Albert Sickler

ZX81

praktische tips programma's BASIC

Kluwer Technische Boeken

ZX 81

Albert Sickler

ZX 81

praktische tips, programma's, BASIC

KLUWER TECHNISCHE BOEKEN B.V. - DEVENTER - ANTWERPEN

Lay-out: Wim Wijnolts
Illustraties: Willem Niessink

ISBN 90 201 1515 4
D/1981/0108/190

© 1981 Kluwer Technische Boeken B.V. Deventer

1e druk 1981

Niets uit deze uitgave mag worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze ook, zonder voorafgaande schriftelijke toestemming van de uitgever.

No part of this book may be reproduced in any form, by print, photoprint, microfilm or any other means without written permission from the publisher.

Ondanks alle aan de samenstelling van de tekst bestede zorg, kan noch de redactie noch de uitgever aansprakelijkheid aanvaarden voor eventuele schade, die zou kunnen voortvloeien uit enige fout, die in deze uitgave zou kunnen voorkomen.

Woord vooraf

De ZX 81 zal ongetwijfeld zijn weg in de Nederlandstalige gebieden vinden. Stellig zal er dan ook de behoefte bestaan aan Nederlandse documentatie voor dit apparaat.
Ik heb me bij de opzet van dit werk vooral door de gedachte laten leiden dat de ZX81 voor velen de eerste computer zal zijn. Sommige zaken dienen dus zeer voorzichtig aangepakt te worden. Dit geldt met name voor de kleine introductie van hardware en software, de introductie in het gebruik van de ZX 81 en de inleiding BASIC.
In de appendices wordt een en ander nog eens gecomprimeerd samengevat. Een aantal appendices (bijv. foutboodschappen) zijn volledig gebaseerd op de enige formele bron nl. de Engelse gebruiksaanwijzing 'ZX 81 BASIC programming' (overigens een uitstekend boek).
Ik hoop dat dit werk in een duidelijke behoefte voorziet.

Albert Sickler

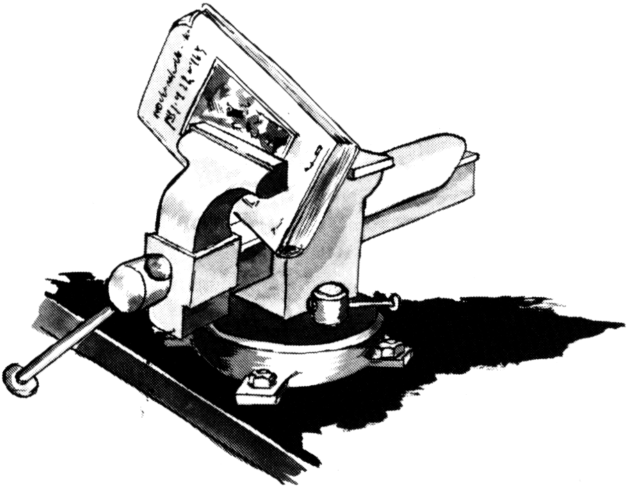
Inhoud

1. Hardware en software 9
2. Aansluiten en aanzetten 14
3. Eerste BASIC-programma 16
4. Het eerste BASIC-programma wordt uitgevoerd 22
5. Algemene procedure voor het opstellen van een programma .. 28
6. Herstellen van fouten 33
7. Het gebruik van de computer als calculator 38
8. De LET-instructie 41
9. REM, PRINT en INPUT 43
10. TAB, AT, CLS en SCROLL 49
11. Over getallen 52
12. Functies met betrekking tot getallen 57
13. IF ... THEN, GOTO, STOP, CONT, BREAK 61
14. FOR ... NEXT-instructie 69
15. Arrays 74
16. Subroutines 79
17. Over letters en andere tekens 83
18. Over plaatjes 94
19. Cassetterecorder en printer 106
20. Machine-instructies 109

Appendix 1: Foutboodschappen 113
Appendix 2: Korte samenvatting van de functies 117
Appendix 3: Korte samenvatting van speciale symbolen 121
Appendix 4: Prioriteitsregels 123
Appendix 5: Instructies en commando's 125
Appendix 6: Symbolen, schrifttekens en codes 132

Programma's 135
Trefwoordenlijst 168

Hoe gebruiken we dit boek?



Zo niet, in ieder geval

De meeste kopers van de ZX81 zullen nog vrijwel geen ervaring hebben op het gebied van computers. Sterker nog ... in veel gevallen zal men deze computer juist willen aanschaffen om deze boeiende ervaring op te doen. De ZX81 is in feite een zeer aantrekkelijk apparaat: voor zeer weinig geld worden we in staat gesteld om zelf programma's te schrijven en uit te voeren.
Gezien deze achtergrond hebben we een boek samengesteld, dat met name op de beginner is gericht. We starten met een bescheiden gehouden inleiding over de begrippen hardware en software, waarbij dan meteen enkele karakteristieken van de ZX81 naar voren komen. Bovendien zal in dit gedeelte bekeken worden waarmee de ZX81 kan worden uitgebreid.
Hierna volgt een volledige BASIC-cursus die uiteraard geënt is op de mogelijkheden van de ZX81. Voor het grootste gedeelte komt deze BASIC-programmeertaal met de gebruikelijke standaard overeen, maar behalve enkele (kleine) afwijkingen zijn er tevens een aantal interessante uitbreidingen.
Blader het boek maar eens door zodat u alvast wat vertrouwd raakt met de verschillende zaken die zoal de revue zullen passeren.
De voorbeelden zijn zo eenvoudig mogelijk gekozen; op deze wijze zien we zo treffend mogelijk wat het effect van een bepaalde constructie is. Enkele voorbeelden zijn zodanig bewerkt dat men al doende ook oog

krijgt voor het opzetten van 'goede' programma's.
Aan het slot van dit boek vindt men praktische informatie o. a. over de te gebruiken grafische symbolen en over de codes voor de zgn. foutboodschappen.
Bij een praktische handleiding horen uiteraard ook praktische wenken. Misschien is de belangrijkste raadgeving wel eerst een gedeelte van dit boek door te nemen (zeker de eerste 6 hoofdstukken) alvorens maar willekeurig op de toetsen te gaan drukken.
Het is niet zo dat de ZX81 daar niet tegen zou kunnen, maar vooral om het feit dat het zomaar drukken op toetsen en kijken wat er gebeurt snel leidt tot 'ik snap er niets van ... veel te ingewikkeld voor mij!'.
Ondanks de vele tekens op het toetsenbord, die waarschijnlijk meer afschrikken dan aanmoedigen, stellen we met nadruk 'het werken met de ZX81 is niet moeilijk!'.
We hopen dat een ieder zeer veel plezier zal beleven aan het programmeren op de ZX81. Programmeren kan behalve als een nuttig ook als een aangenaam tijdverdrijf worden ervaren.
Ten slotte willen we hier vermelden dat dit boek tevens kan worden gebruikt bij de ZX80 die dan wel voorzien moet zijn van de uitgebreide BASIC-versie.

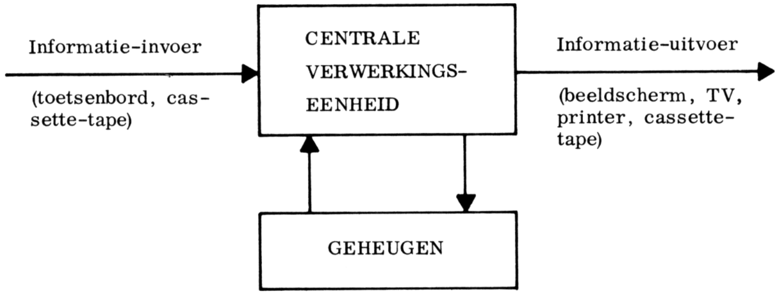
1. Hardware en software

Inleiding

Het Engelse werkwoord 'to compute' betekent 'berekenen'. Hieruit zou men wellicht kunnen afleiden dat een computer zoiets als een rekenapparaat is. De term rekenen doet echter te kort. Een computer kan veel meer dan enkel maar rekenen.
Alles draait om het begrip informatie, want een computer is met name een apparaat dat informatie volgens een opgegeven rij instructies kan bewerken.
De informatie die bewerkt moet worden kan betrekking hebben op getallen, maar evenzo kunnen we bij informatie denken aan bepaalde patronen (bijvoorbeeld bepaalde figuren zoals die op een TV-scherm kunnen worden afgebeeld), aan letters (denk eens aan alle toepassingen voor tekstverwerking), aan muziektonen enz.
Bij de instructies moeten we niet alleen denken aan rekenkundige bewerkingen zoals optellen, aftrekken, vermenigvuldigen en delen,maar ook aan instructies zoals 'maak het beeldscherm schoon', 'plaats de letter A vooraan de regel op het beeldscherm' enz.
Bij de computer worden gewoonlijk de begrippen hardware en software onderscheiden. In het nu volgende zullen we dieper op deze begrippen ingaan. We zullen zien dat de hardware met het werkelijk tastbare aan de computer (componenten, toetsenbord enz.) te maken heeft, terwijl de software te maken heeft met de instructies die we de computer kunnen opgeven, de zgn. programma's.

Hardware

Een computer, kortom een informatiebewerker, kunnen we als volgt in beeld brengen:



Figuur 1.1 Algemene voorstelling van de computer

In dit schema zien we twee blokken die respectievelijk de namen 'centrale verwerkingseenheid' en'geheugen' dragen.
De centrale verwerkingseenheid en het geheugen vallen onder de zgn. hardware van de computer. Onder de hardware zullen we alle tastbare zaken van de computer vestaan, waarbij het woord tastbaar in dit geval letterlijk genomen moet worden. Elektronische componenten, toetsenbord en beeldscherm zijn voorbeelden van tastbare zaken en vallen dus onder de hardware.

De centrale verwerkingseenheid (Engels: central processing unit, of afgekort CPU) vormt het feitelijke hart van onze computer. Deze eenheid voert de opgedragen bewerkingen (instructies) uit. We moeten ons daarbij voorstellen dat er reeksen instructies zijn, die in het geheugen zijn opgeslagen. Deze worden dan een voor een uit het geheugen uitgelezen en uitgevoerd.
Aangezien een geheugen alleen maar reeksen enen en nullen kan bevatten moeten we er wel van uit gaan dat ook de instructies door middel van enen en nullen zijn gecodeerd. Dit is inderdaad zo ... wanhoop echter niet, de gebruiker wordt in staat gesteld om instructies in 'zeer verstaanbare taal' op te geven.
We moeten ons echter wel bedenken dat iedere toetsindruk (op het toetsenbord) elektronisch in een rijtje enen en nullen wordt omgezet. Zo zullen uiteindelijk ook termen afkomstig uit onze nog te bespreken programmeertaal in rijtjes enen en nullen worden omgezet. Dat wil zeggen dat onze computer zo is ingericht dat de instructies (afkomstig uit de Engelse taal, zoals PRINT, d.w.z. 'druk af', INPUT, d.w.z. 'voer in') in de juiste serie enen-en nulleninstructies worden omgezet. Deze enen- en nulleninstructies leiden dan tot het beoogde resultaat.
De ZX81 werkt met een zeer bekend type CPU, nl. de centrale verwerkingseenheid die wordt aangeduid met het type Z80 (A). Over dit type zijn reeds vele, meest Engelstalige, boeken geschreven.
Een en ander is met name van belang als we ons bezig willen houden met het programmeren in instructies die niet eerst behoeven te worden omgezet, maar die dan direct door de centrale verwerkingseenheid worden begrepen.
We doelen hier op de zgn. machinecodes (of machine-instructies). Voor een beginner is dit echter vrij onverteerbare stof, vandaar ons advies: leer eerst BASIC! In veel gevallen is deze taal toereikend voor al uw problemen.

In het schema zien we bij de term invoer de woorden: toetsenbord en cassettetape. Hiermee worden de twee mogelijkheden aangegeven om onze computer van informatie te voorzien.
Onder informatie verstaan we hier: de gegevens die bewerkt moeten worden, alsmede de reeksen instructies die aangeven wat er precies

moet gebeuren. Een dergelijke reeks instructies noemt men trouwens 'een programma' en het opstellen van een dergelijke reeks geven we met de term'programmeren'aan.
Het zal duidelijk zijn dat we informatie kunnen invoeren door op verschillende toetsen van het toetsenbord te drukken.
Als we de ZX 81 bekijken zal in ieder geval opvallen dat het toetsenbord tamelijk afwijkend is van het toetsenbord van bijv. een schrijfmachine. Vooral het grote aantal termen en symbolen valt op. Denk nu niet 'dit is te ingewikkeld...', we komen er nog uitgebreid op terug en dan zal blijken hoe eenvoudig alles in zijn werk gaat.
Het gebruik van de cassetterecorder ligt minder voor de hand als medium voor invoer. Ook dit onderwerp komt nog uitgebreid ter sprake. We merken hier reeds op dat cassettes vooral gebruikt worden om afzonderlijke programma's op te kunnen slaan en als zodanig ook in te kunnen voeren. De firma SINCLAIR levert reeds een aantal in feite 'kant en klaar'-programma's.
In het schema zien we bij de term uitvoer de woorden: beeldscherm (TV), printer en cassettetape. Hiermede worden dus de mogelijkheden opgesomd om informatie uit te voeren.
In ieder geval is er een beeldscherm nodig. Op het beeldscherm kunnen allerlei resultaten van bijvoorbeeld berekeningen worden weergegeven maar ook zgn. 'animated graphics' (aantrekkelijke plaatjes die o. a. bij spelletjes worden gebruikt).
Als beeldscherm gebruiken we een gewone TV (zwart/wit of kleur).
We zullen zien dat de ZX 81 op twee manieren kan werken, nl. op een relatief trage wijze 'slow mode' en op een relatief snelle wijze 'fast mode'. Bij de eerstgenoemde wijze blijft het beeldscherm voortdurend oplichten terwijl bij de tweede wijze het beeldscherm dooft tijdens het rekenen. We zouden kunnen zeggen: het apparaat rekent zo snel dat het geen tijd heeft om tussenresultaten te tonen. Dit is inderdaad letterlijk zo.
De ZX 81 is zo ingericht dat het beeldscherm van de TV is onderverdeeld in:

24 horizontale regels die meestal met 'lijnen' worden aangeduid.

en 32 gedeelten per horizontale lijn.

De onderste twee horizontale lijnen zijn steeds gereserveerd om er bepaalde zaken op zichtbaar te maken. De rest, dat wil zeggen de bovenste 22 horizontale lijnen (ieder onderverdeeld in 32 partjes) kunnen geheel volgens uw wens met tekens (schrifttekens, cijfers en grafische symbolen) worden gevuld. Hoe men dit kan doen zullen we nog uitgebreid bespreken.

We kunnen informatie ook weer op cassettetape opslaan. Ten slotte kan informatie worden afgedrukt op een zgn. printer. Met het woord printer wordt 'afdrukapparaat' bedoeld. Het woord is echter zo ingeburgerd

dat een ieder nu spreekt van een printer. Het grote voordeel van een printer is dat we afdrukken van resultaten hebben die ook als zodanig bewaard kunnen worden. Zo kunnen we bijvoorbeeld in alle rust de resultaten van een bepaald programma doornemen om na te gaan of alles wel volgens de verwachting is verlopen. De firma SINCLAIR heeft voor de ZX81 een afzonderlijke printer ontworpen. Alleen deze printer kan aan de ZX81 worden gekoppeld!

De centrale verwerkingseenheid wisselt voortdurend informatie met het geheugen uit. Dat wil zeggen: er worden rijtjes enen en nullen vanuit het geheugen opgehaald (uitleesactie) en er worden rijtjes enen en nullen op bepaalde plaatsen in het geheugen opgeslagen (inleesactie). Opdat de computer goed kan functioneren dient althans een deel van het geheugen dus een bijzondere eigenschap te hebben. Deze eigenschap houdt in dat op willekeurige plaatsen van het geheugen zowel reeksen enen en nullen (informatie) moeten kunnen worden in- en uitgelezen. Deze eigenschap wordt met de term RAM (Engels: Random Access Memory) aangegeven. De ZX81 beschikt standaard over een RAM-geheugen dat als een opbergkast is onderverdeeld in 1024 'laden'. Iedere lade bevat 8 hokjes waarin men de tekens 1 of 0 kan opslaan. Doorgaans spreken we niet van laden maar van woorden.
In het gebruikelijke computertaaltje omschrijven we dit als 1K (=1024) woorden (= laden) van 8 bits (= enen en/of nullen). De grootheid 8 bits wordt ook wel eens met de term byte aangegeven. Dit is niet zo veel. De serieuze gebruiker zal zich al snel gedwongen zien om de afzonderlijke geheugen-uitbreidingsmodule die SINCLAIR levert aan te schaffen (16K - RAM - uitbreidingsmodule).
Het standaard aanwezige RAM-geheugen is net genoeg om ervaring met de verschillende instructies op te doen. De programma's die in het standaardgeheugen passen zijn doorgaans maar enkele tientallen regels lang (een en ander hangt natuurlijk wel af, van wat zo'n programma doet, bijv. of er veel variabelen voorkomen, of er veel tekst op het beeldscherm moet komen enz.).
De voorbeeld-programma's in dit boek zijn zo gekozen dat ze met de standaard aanwezige geheugenruimte kunnen worden uitgevoerd.

Behalve RAM-geheugen beschikt de ZX81 nog over ROM-geheugen. Dit geheugen heeft de bijzondere eigenschap dat we er alleen maar informatie uit kunnen lezen (en er dus niets in kunnen opslaan). Vandaar de naam ROM, dat wil zeggen Read Only Memory, hetgeen vertaald betekent: Lees Alleen Geheugen.
In dit geheugen heeft de fabrikant al kant en klare programma's opgeslagen, onder andere het programma dat de door u ingetoetste BASIC-instructies vertaalt in de machinecodes die leiden tot het beoogde resultaat. Dit in ROM opgeslagen programma wordt de 'BASIC-interpreter' genoemd.

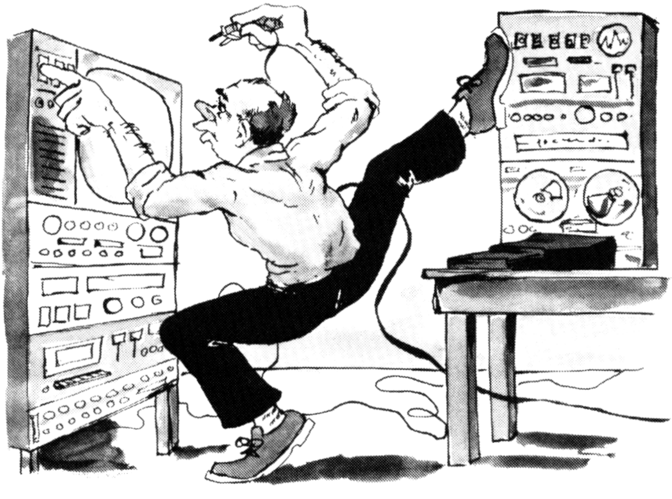
Software

Als we het hebben over programma's die in ROM zijn opgeslagen, als we het hebben over de BASIC-interpreter, als we het hebben over macinecodes, dan hebben we het over software.
Software is een verzamelnaam voor alles wat met programma's te maken heeft. Men stelt wel eens dat de software de ware kracht van een computer bepaalt. In zekere zin is dit volstrekt waar. De ZX 81 is met name een aantrekkelijke machine omdat we hier onmiddellijk 'krachtige' BASIC-programma's op kunnen verwerken.
Beschouwen we de software dan kunnen we ruwweg drie categorieën herkennen.

a. Alle programma's die te maken hebben met het functioneren van de machine zelf. Dit zijn de programma's die de fabrikant reeds in het ROM-geheugen heeft opgeslagen.
b. Programma's die door de gebruiker zelf kunnen worden opgesteld.
c. Programma's die men kant en klaar kan kopen en onmiddellijk kan verwerken.

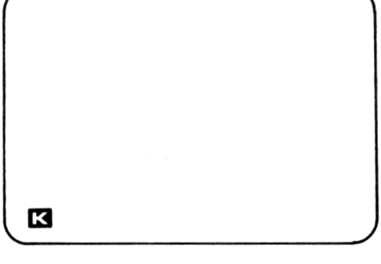
We starten met categorie a. Alle programma's die met het functioneren van de machine zelf te maken hebben duidt men wel eens aan met de naam 'operating system'. Bij de wat kleinere machines zoals de ZX 81 ziet men ook wel eens de naam 'monitor'. We zullen er op dit moment nog niet verder op ingaan. Categorie b is voor de gebruiker van de ZX 81 van groot belang. In de meeste gevallen zal men de ZX 81 immers aanschaffen om ervaring in het programmeren op te doen. De belangrijkste taal om instructies aan te geven is BASIC. BASIC vormt dan ook de hoofdmaaltijd van dit boek. De gebruiker wordt echter ook in staat gesteld om machinecodes te gebruiken. We zullen hier alleen maar even bij blijven stilstaan omdat het in feite buiten ons gezichtsveld valt.
Categorie c kan ook van belang zijn. Meer en meer programma's worden kant en klaar op cassette geleverd. Bedenk dat alleen cassettes waarop uitdrukkelijk vermeld staat dat ze voor uw ZX 81 geschikt zijn, als zodanig gebruikt kunnen worden. Het heeft dus geen enkele zin om cassettes aan te schaffen waarop staat, dat ze voor een andere computer zijn ontwikkeld.
Het is raadzaam om van het begin af aan alle bruikbare programma's op cassette op te slaan en wel voorzien van goede documentatie. Aan dat laatste wil het nog wel eens ontbreken.

2. Aansluiten en aanzetten



Alvorens we de ZX 81 kunnen gebruiken moet deze op de juiste wijze worden aangesloten. Voer daartoe de volgende handelingen uit:

1. Verbind de ZX 81 met een TV (zwart/wit of kleuren). Voor deze verbinding is een afzonderlijk snoer meegeleverd. Een uiteinde past in de antenne-ingang van de TV en het andere uiteinde past op de connector (verbindingsstuk) op de ZX 81 waar 'TV' bij staat.
2. Verbind het snoer van het voedingsapparaat (doosje met daaraan gekoppelde stekker voor 220 V) aan de ZX 81 en wel door de kleine stekker van dit snoer in de connector waarbij '9 V DC in' staat, te steken.
3. Zet de TV aan en stem af op kanaal 36 (UHF).
4. Steek de stekker van het voedingsapparaat in een stopcontact.
5. Draai nu net zolang aan de afstemknop van de TV tot het onderstaande beeld duidelijk waarneembaar is.



De aansluiting van de cassetterecorder en de aansluiting van de printer worden in hoofdstuk 19 besproken.
Indien we de beschikking hebben over een RAM-module (geheugenuitbreiding) dan schuiven we deze module (met de voedingsspanning uit!) in de daartoe aanwezige gleuf. Dit kan maar op een manier ... dus geen problemen.
Bedenk dat bij het uitzetten van de computer (trek stekker van voedingsapparaat uit stopcontact) alle informatie verloren gaat, dat wil zeggen ook het programma dat u misschien net te voren had ontwikkeld. Reden genoeg om tijdens het zelf ontwikkelen van programma's de cassetterecorder te gebruiken, hierop kunnen we tenminste programma's voor permanent gebruik vastleggen.
Van het feit dat bij het weer aanzetten van de computer alles 'schoon' is, wordt ook wel eens met opzet gebruik gemaakt. Als u volledig vast zit, dat wil zeggen er gebeuren dingen die u volstrekt niet meer kunt verklaren en het apparaat reageert ogenschijnlijk zeer vreemd op uw commando's; zet dan het apparaat uit en vervolgens weer aan. Uiteraard begint u dan met een schone lei, maar de ZX81 zal nu wel reageren zoals het hoort.

3. Eerste BASIC-programma

Het zou nu redelijk zijn om het toetsenbord van de ZX81 te behandelen. De meesten zullen bij het aanschouwen van alle symbolen bij deze toetsen wellicht denken 'hoe krijgen we dat nu onder de knie?'. We willen hier al direct vermelden dat een en ander in de praktijk geen enkel probleem zal opleveren. Toch is het niet verstandig om op dit moment al alle zaken betreffende het toetsenbord uit te leggen. Voor het juiste begrip is nl. enige kennis omtrent BASIC nodig.
Deze kennis zal in dit gedeelte worden aangedragen. Daarom bevelen we aan het nu volgende eerst goed door te lezen. In het vervolg zullen we dan zien hoe programma's kunnen worden ingetoetst.

> EXTRA WAARSCHUWING: Lees dit hoofdstuk alleen door en toets nog niets in. Het intoetsen wordt in het volgende hoofdstuk besproken.

Programmeren ... het woord roept bij menigeen nog bepaalde associaties op. Programmeren zou bijv. het voorrecht zijn van slechts enkele wiskundig onderlegde personen ...
In het nu volgende zullen we tonen dat programmeren in het geheel niet moeilijk is en dat iedereen plezier aan het programmeren kan beleven.
Een programma is een reeks instructies. Als de computer de opeenvolgende instructies uitvoert dient het beoogde resultaat te verschijnen.
Bij de programmeertaal BASIC worden de instructies op afzonderlijke

regels geschreven. Voor iedere regel dient men een regelnummer te plaatsen. Een regelnummer dient steeds een positief getal te zijn. Deze regelnummers dienen tevens oplopend in grootte te zijn. Uitdrukkelijk stellen we dat de nummers elkaar niet precies hoeven op te volgen, zoals dat bijvoorbeeld wel het geval is bij de reeks 1, 2, 3, 4, enz. Dit is overigens wel een correcte reeks regelnummers, maar deze reeks biedt geen enkele mogelijkheid om naderhand nog instructies in te voegen (dit zal nog besproken worden).
Nummeren we daarentegen met de nummers 10,20,30,40 enz. dan kan men eventueel tussen iedere regel nog 9 andere regels voegen.
Laten we voorlopig maar afspreken dat we om deze reden in onze voorbeeldprogramma's steeds volgens de reeks 10,20,30,40 enz. de regels van de programma's zullen nummeren. De nummering van de regels zal om twee redenen van groot nut blijken te zijn:

- bij het opmaken c.q. wijzigen van een programma kan men handig van deze nummers gebruik maken;
- het zal blijken dat we in programma's sprong-instructies kunnen opgeven. Bij deze sprong-instructies dienen de regelnummers als het ware als labels (verwijs-adressen) waar naar toe gesprongen moet worden.

In het nu volgende tonen we een programma dat de rente-opbrengst na een jaar, van een bepaald kapitaal nl. ƒ 1100,- uitrekent. Bij dit voorbeeld is als rente-percentage 8% genomen.
Voor alle duidelijkheid, de rente-opbrengst vinden we door het kapitaal met 0.08 te vermenigvuldigen.
Het resultaat (berekende rente-opbrengst) dient op het beeldscherm te worden 'afgedrukt' ('geprint').

Hier volgt het volledige programma:

```
10 LET KAP=1100
20 LET RENTE=0.08
30 LET BEDRAG=KAP*RENTE
40 PRINT BEDRAG
```

Als resultaat van dit programma dient het getal 88 op het beeldscherm te verschijnen.

We zien dat iedere regel van dit programma inderdaad met een nummer begint en bovendien dat we onze conventie hebben aangehouden door te nummeren volgens 10,20,30 en 40.
De eerste regel luidt 'vrij vertaald':

10 laat KAP gelijk aan 1100 zijn

Met andere woorden, in deze regel (instructie) wordt de waarde 1100

(denk maar aan f 1100,-) aan KAP toegekend. We moeten ons daarbij voorstellen dat de computer onmiddellijk na het 'lezen' van deze instructie een stukje geheugenruimte reserveert en deze geheugenruimte dan van het label KAP voorziet. In plaats van geheugenruimte hanteert men ook wel eens het begrip 'doos' en men moet dan denken aan een doos voor de opslag van getallen. Volgens deze eerste instructie van ons BASIC-programma dient in de doos met het opschrift KAP de waarde 1100 te worden opgeslagen. Men had natuurlijk ook wel een andere waarde in KAP kunnen opslaan en om dit karakter aan te geven (een variabele grootheid kan worden opgeslagen) spreekt men in het computerwereldje niet van doos maar van 'variabele', in dit geval van de variabele KAP.
We weten dus dat een variabele niet iets moeilijks voorstelt, maar gewoonweg een doos voor de opslag van een getal: een doos die dan voorzien is van een naam.
Elke variabele (= elke doos) draagt in BASIC een naam. Bij de ZX81 heeft men een grote vrijheid ten aanzien van het geven van namen.
De regel met betrekking tot het geven van een naam luidt:

> een naam bestaat steeds uit letters (inclusief spaties) en cijfers waarbij het eerste schriftteken altijd een letter moet zijn.

We merken hierbij op dat de regels bij de ZX81 veel ruimer zijn, dan de standaardregels die doorgaans worden gehanteerd.
Voorbeelden van correcte namen zijn:

```
RENTE

RENTE BEDRAG

X

ONBEKENDE 1

ZX81
```

Voorbeelden van incorrecte namen zijn:

`4KEER` (begint met cijfer)

`NOU?` (vraagteken is niet toegestaan)

De tweede regel van het voorbeeldprogramma luidt:

```
20 LET RENTE=0.08
```

Deze regel zal nu wel geen problemen opleveren: er wordt mee aangegeven dat aan de variabele met de naam RENTE het getal 0.08 moet worden toegekend.
Merk op dat de 'komma' in een getal hier door middel van een punt

wordt aangegeven. Deze gewoonte komt overeen met het Engelse gebruik, waar men ook spreekt van de 'decimal point' (decimale punt). De derde regel luidt:

```
30 LET BEDRAG=KAP*RENTE
```

Deze regel geeft aan dat aan de variabele met de naam BEDRAG de waarde moet worden toegekend die men krijgt door KAP met RENTE te vermenigvuldigen. Merk op dat het vermenigvuldigteken hier met het sterretje * wordt aangegeven. Dit gebruik voorkomt mogelijke verwarring met de letter x. Om eenzelfde reden plaatst men vaak een schuin streepje door het cijfer O, met andere woorden Ø om zo verwarring met de letter O te voorkomen. Als gevolg van de getoonde instructie krijgt de variabele met de naam BEDRAG de waarde 1100 x 0. 08 = 88.

ONTHOUDT VOOR ALTIJD:

- Het cijfer O wordt op het toetsenbord aangegeven met Ø
- Het vermenigvuldigteken wordt aangegeven met de ster *
- De komma in een getal wordt altijd aangegeven met een punt.

De daarop volgende regel:

```
40 PRINT BEDRAG
```

bevat een print-instructie. Deze instructie geeft aan dat de waarde van BEDRAG op het beeldscherm moet worden afgedrukt.
Dit was de laatste instructie van het programma. Bij de meeste BASIC-versies dient men als laatste instructie nog de END-instructie op te nemen, bijvoorbeeld:

```
50 END
```

De BASIC-versie van de ZX81 staat de gebruiker echter toe om de laatste END-instructie weg te laten (sterker nog: het is bij de ZX81 niet eens mogelijk).

Op dit moment kunnen we al een belangrijke raadgeving ten aanzien van het opstellen van correcte programma's geven.
Deze raadgeving houdt in, zodanige namen te kiezen dat de lezer van het programma zo goed mogelijk kan begrijpen waar een en ander om gaat.
Ter illustratie: het volgende programma heeft hetzelfde effect als het voorafgaande programma:

```
1 LET A=1100
23 LET B3=0.08
47 LET C7B=A*B3
48 PRINT C7B
```

De namen zijn hier echter, evenals de nummers van de regels,zeer willekeurig gekozen. De lezer die het programma voor de eerste keer leest zal geen enkele achtergrond over het programma kunnen proeven. De regelnummering is zonder meer vreemd zodat men snel geneigd zal zijn er iets achter te zoeken.

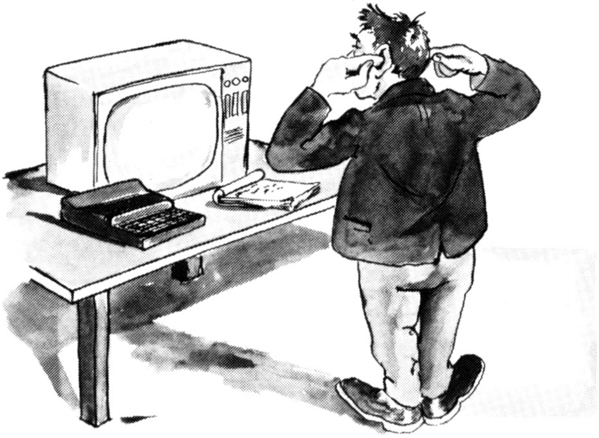
We plaatsen bij het geven van namen nog twee opmerkingen. De eerste opmerking betreft in feite een reeds eerder gemaakte opmerking, nl. dat de vrijheden van de ZX81 voor het geven van namen beduidend groter zijn dan bij de meeste andere BASIC-versies. Dit heeft dan voornamelijk betrekking op het feit dat bij deze versies minder ruimte bestaat voor het geven van suggestieve namen.
De tweede opmerking heeft betrekking op de hoeveelheid RAM-geheugen (voor programmeren beschikbare geheugen) die bij de ZX81 standaard aanwezig is. Deze geheugenruimte is, zoals gezegd, niet erg groot. Bedenkt men dat iedere letter van een naam een vakje (van 8 bits) beslaat, dan zal een programma met een relatief groot aantal variabelen dit geheugen al snel consumeren. In de praktijk zal men dus vaak korte namen kiezen (zo kozen we voor de naam KAP in plaats van de wellicht meer voor de hand liggende naam KAPITAAL).
Beschikt men over de module om het geheugen uit te breiden, dan adviseren we uiteraard om namen volgens de gegeven richtlijnen te kiezen.

In dit gedeelte zullen we nog niet uit de doeken doen hoe men het programma moet intoetsen om het vervolgens ook werkelijk door de computer te laten uitvoeren.
Voorlopig volstaan we met de opmerking dat het om een volledig (correct) programma gaat, dat in het geheugen van de computer kan worden opgeslagen om dan vervolgens na het geven van een zeker commando te worden uitgevoerd.
De computer zal na dit commando instructie voor instructie uitvoeren. Zo zal als gevolg van de eerste instructie geheugenruimte worden gecreëerd en deze ruimte wordt voorzien van het label KAP. In deze ruimte wordt in de binaire representatie (representatie volgens bepaalde afspraak met de tekens 1 en 0) het getal 1100 opgeslagen. Hierna wordt de ruimte RENTE gereserveerd waarin het getal 0.08 wordt opgeslagen. Vervolgens wordt de ruimte GETAL gereserveerd, de waarde 1100 x 0.08 wordt uitgerekend en de uitkomst hiervan wordt in GETAL geplaatst.
Ten slotte wordt de inhoud van de geheugenruimte met het label GE-

TAL door middel van de PRINT-instructie op het beeldscherm afgebeeld. Niet als een serie enen en nullen (in feite de wijze waarop het getal wederom in binaire vorm is opgeslagen) maar in direct herkenbare cijfers uit het ons zo vertrouwde tientallige stelsel. Er zijn afzonderlijke voorzieningen getroffen die tot effect hebben, dat we op het beeldscherm geen woud van enen en nullen zien, maar gewone herkenbare cijfers.

In het programma worden cijfers gebruikt (voor het aangeven van regelnummers en getallen) speciale woorden (LET en PRINT) die een gereserveerde betekenis in BASIC hebben en namen die door middel van letters en cijfers worden gespeld.
Deze opmerking is vooral van belang met betrekking tot het nu volgende hoofdstuk waarin getoond wordt hoe het programma wordt ingetoetst en uiteindelijk ook door de computer wordt uitgevoerd.

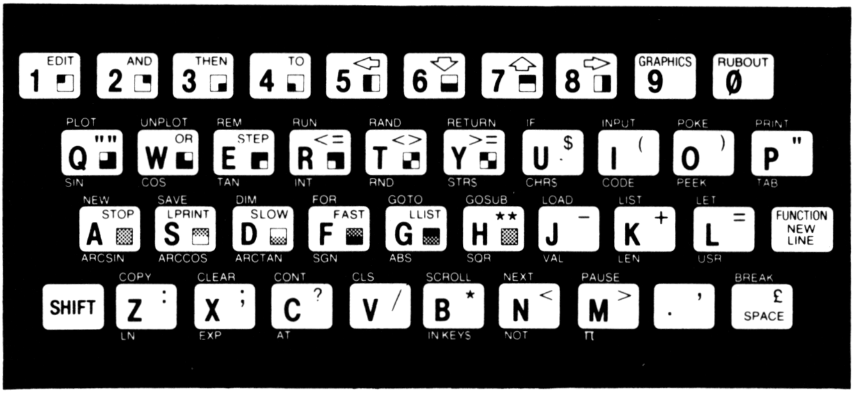
4. Het eerste BASIC-programma wordt uitgevoerd



In het nu volgende gedeelte zullen we tonen hoe het programma over de rente-berekening op de ZX 81 kan worden uitgevoerd.
Staat de computer al aan ... ? Zo niet, voer dan de in hoofdstuk 2 gegeven instructies uit. In de linker benedenhoek zien we de letter K in de 'inverse video'-wijze oplichten, dat wil zeggen in een zwart blokje zien we de letter K in wit geschreven.
We brengen nu nogmaals het programma in herinnering:

```
10 LET KAP=1100
20 LET RENTE=0.08
30 LET BEDRAG=KAP*RENTE
40 PRINT BEDRAG
```

Het zal duidelijk zijn dat dit programma eerst zal moeten worden ingetoetst.
De ZX 81 is uitgerust met een soort tiptoetsen: aanraken is echter niet genoeg, de toetsen moeten even worden ingedrukt.
Bij sommige toetsen komen wel vijf symbolen voor, dus al snel rijst de vraag hoe we met dit toetsenbord moeten omgaan.
Op de volgende bladzijde zien we een overzicht van het volledige toetsenbord.
Merk allereerst op dat iedere toets een 'hoofdsymbool' bevat. Daar-



naast zien we in veel gevallen grafische symbolen: vierkantjes die op een bepaalde wijze gevuld zijn. Deze grafische symbolen worden alleen opgeroepen als van te voren bepaalde toetsen zijn ingedrukt. We zullen dit in hoofdstuk 17 behandelen.
Daarboven zien we symbolen die in de kleur rood zijn weergegeven (bijv. boven de toets met hoofdsymbool A zien we in rood STOP aangegeven). De term symbool is in dit verband misschien wat misleidend want vaak zien we complete termen. Ter wille van de eenvoud zullen we alle termen die we bij de toetsen zien ook maar als symbool aanduiden.
De symbolen die in de rode kleur zijn aangegeven worden opgeroepen als tevens de toets SHIFT, die ook in rood is aangegeven, ingedrukt gehouden wordt. Deze handelwijze stemt volledig overeen met het gebruik bij een gewone schrijfmachine.

> Vergeet nooit dat bij het intoetsen van een rood aangegeven symbool de toets SHIFT van te voren moet worden ingedrukt en hierna ook ingedrukt gehouden moet worden.

Probeer het nu echter nog niet uit, want u zou nog voor vreemde verrassingen kunnen komen te staan.
Dan resten ons nog de symbolen boven en onder de toetsen. De eerste groep (symbolen boven de toetsen) bestaat voornamelijk uit specifieke termen die in BASIC worden gebruikt. Zo kwamen we in ons programma de specifieke termen LET en PRINT tegen. We merken hierbij tevens op, dat na ieder regelnummer in een BASIC-programma altijd zo'n specifieke term volgt.

Als de computer zo'n specifiek woord verwacht wordt dit aangegeven door de letter K in het vlakje in de linker benedenhoek van het beeldscherm. De letter K komt van Keyword, dat hier sleutelwoord of meer in het algemeen 'specifieke term' betekent. Men gebruikt ook wel eens de uitdrukking 'gereserveerd woord'.
De letter in het zwarte hokje noemen we kortweg 'de cursor'. (Vergelijk de cursor maar met een aanwijsstok.)
Als de computer een term als LET of PRINT, met andere woorden zo'n specifieke term, verwacht en u drukt op een toets waarbij zo'n term staat (zo staat bij de letter L de term LET) dan wordt die term opgeroepen, dat wil zeggen deze term verschijnt als geheel op het beeldscherm. We zullen zo dadelijk zien hoe dit aan de hand van het voorbeeldprogramma verloopt.

Ten slotte zien we nog symbolen en termen onder de toetsen. Deze symbolen hebben betrekking op voorgeprogrammeerde functies. Deze functies (die trouwens nog afzonderlijk behandeld zullen worden) kan men intoetsen door eerst FUNCTION in te toetsen (d.w.z. door op SHIFT en NEWLINE te drukken). Na het op de aangegeven wijze intoetsen van FUNCTION verandert het symbool van de cursor in de letter F ten teken dat de computer nu de naam van een functie verwacht. Drukt men hierna op een toets waarbij een aanduiding onder de toets staat, dan verschijnt die aanduiding op het beeldscherm.
Ziezo, dat was de inleiding, nu dan het feitelijke intoetsen van ons voorbeeldprogramma.
We drukken allereerst op de toetsen 1 en 0 (deze vormen immers het eerste regelnummer van ons programma). Vervolgens moet LET worden ingetoetst. We drukken daartoe op de toets waarboven LET staat (toets met letter L). De computer toont nu inderdaad LET op het beeldscherm en wel inclusief de spatie tussen 10 en LET. Hier wordt LET getoond omdat na 10 een specifieke term moet komen en de letter L zelf is nu eenmaal geen specifieke term.
Nu verandert het symbool van de cursor in de letter L: hiermee wordt aangegeven dat de computer nu een Letter (of schriftteken) verwacht.
We toetsen vervolgens in:

K A P

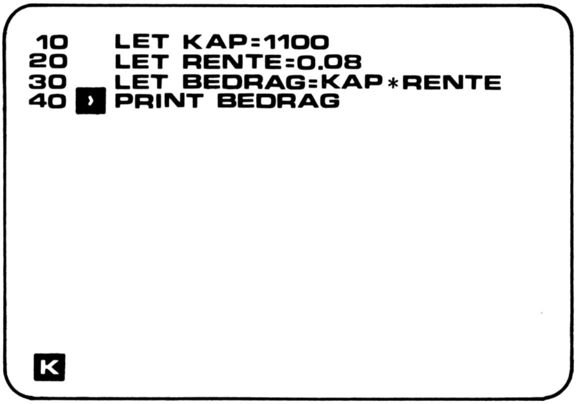
Het symbool = kunnen we intoetsen door op SHIFT en L (bij deze toets zien we immers het symbool =) te drukken, waarbij we natuurlijk niet vergeten tijdens het intoetsen van L de toets SHIFT ingedrukt te houden.
Hierna toetsen we in:

1 1 0 0

(N.B. het cijfer O wordt door het symbool Ø aangegeven)

Alles goed gegaan ... ? Zo ja, dan drukken we op NEWLINE. De ingetoetste regel wordt nu bovenaan het beeldscherm afgebeeld samen met een tweede cursor nl. een pijltje > achter de 10.
Indien u nu al per ongeluk een fout heeft gemaakt (kortom, er gebeuren 'vreemde' dingen) kunt u het beste het apparaat uit en weer aan zetten om dan vervolgens de besproken procedure te herhalen. * (N. B. een foute toetsaanslag kunnen we onmiddellijk wissen door SHIFT ingedrukt te houden en vervolgens RUBOUT in te toetsen) We zullen in hoofdstuk 6 zien dat er uiteraard nog wel elegantere methoden zijn om fouten te herstellen.
De cursor met het pijltje laten we voorlopig buiten beschouwing.

We vervolgen het intoetsen van het programma. Als gevolg van het indrukken van NEWLINE toont de cursor inmiddels weer een K ten teken dat of een regelnummer of een specifieke term wordt verwacht.
We typen nu in: 2 0 en vervolgens LET door weer op de toets met de letter L te drukken. Hierna wordt ingetoetst: R E N T E SHIFT en = 0 . 0 8 en NEWLINE.
Na het indrukken van NEWLINE zal de zojuist ingetoetste regel onder de vorige regel worden geplaatst.
Na deze uitleg zal het intoetsen van de twee volgende regels wel geen problemen meer opleveren. (vergeet niet dat PRINT weer een gereserveerd woord is, m. a. w. na 40 drukt u slechts enkel op de P)
Merk op dat er bij de gereserveerde woorden zoals LET en PRINT automatisch, zowel er voor als er achter, een spatie wordt geplaatst. Heeft men niet met dergelijke gereserveerde woorden te maken dan moet men eventuele spaties afzonderlijk intoetsen en wel door op de toets met SPACE (rechts onderaan) te drukken. Als alles goed is gegaan toont uw beeldscherm nu het volgende:



* Men kan ook SHIFT ingedrukt houden en dan zo vaak op de toets RUBOUT drukken tot de gehele regel is gewist.

De spanning stijgt ... we zouden het programma nu kunnen uitvoeren. Opdat de ZX81 het programma nu daadwerkelijk gaat uitvoeren moeten we eerst een commando geven.
Een commando verschilt van een instructie op een aantal punten.
Ten eerste typen we nu geen regelnummer in; het commando vormt immers geen instructie in het programma.
Ten tweede: een commando dient direct door de computer te worden uitgevoerd.
Commando's zijn ook weer gereserveerde woorden die men door middel van een enkele toetsindruk activeert (als de cursor tenminste de letter K toont).
De bekendste commando's zijn:

NEW: Als het commando NEW wordt gegeven wordt het geheugen geheel en al gewist. Dit commando kunnen we dus gebruiken als een nieuw programma moet worden ingetoetst. Op dit moment is het onverstandig om dit commando uit te voeren, omdat ons programma dan immers verloren gaat.

LIST: Toont het programma op het scherm. We noemen de weergave van het programma ook wel eens een listing. Als een programma meer regels bevat dan het beeldscherm kan tonen, kunnen we gedeelten op het beeldscherm krijgen door na LIST het regelnummer te toetsen van waaraf het programma getoond moet worden.

RUN: Dit is het commando dat de computer de opdracht geeft het programma uit te voeren ... te 'runnen'. Eventueel kunnen we na RUN nog een regelnummer opgeven om op deze wijze aan te geven dat het programma vanaf dat nummer doorlopen moet worden.

Bedenk dat we ieder commando moeten afsluiten door op NEWLINE te drukken.
Het moment van de waarheid is aangebroken ... we toetsen op (de letter R, d.w.z.):

RUN (niet vergeten hierna op NEWLINE te drukken)

Onmiddellijk verschijnt het antwoord 88 in de linker bovenhoek. In de linker benedenhoek verschijnt de aanduiding:

0/40

Het getal 40 achter de schuine streep geeft het regelnummer aan waarmee het programma werd beëindigd. De 0 geeft aan dat er geen fouten tijdens het uitvoeren van het programma zijn ontdekt, kortom, een geruststellende mededeling.
Drukken we hierna weer op NEWLINE dan verschijnt de listing van het programma weer. We kunnen het programma zo vaak uitvoeren als we maar wensen. Iedere keer na RUN verschijnt het antwoord weer...

enige vermoeidheidsverschijnselen zijn niet waar te nemen.

We zullen dit hoofdstuk afsluiten met het demonstreren van de commando's LIST en NEW.
Allereerst geven we het commando:

```
LIST 30
```

Nu verschijnt het programma vanaf regel 30. Is het eerste gedeelte verloren gegaan?
We toetsen in:

```
LIST
```

en het volledige programma verschijnt weer. U kunt het programma nu bijv. ook weer opnieuw uitvoeren (runnen) door het commando RUN te geven.

Ten slotte het commando NEW. We hebben reeds vermeld dat het resultaat van NEW bestaat uit het wissen van het geheugen ... kortom, het is een commando waarmee u extra voorzichtig moet zijn.
We toetsen in:

```
NEW
```

Het commando LIST toont dat ons voorbeeldprogramma nu inderdaad tot de verleden tijd behoort (we zien nu alleen nog maar de boodschap 0/0 en de listing van het programma is verdwenen).
Het commando NEW was een waardig commando om dit hoofdstuk te besluiten. Het geheugen is schoon. Het vorige programma kan niet meer worden uitgevoerd en het enige dat we nu nog kunnen doen is een nieuw programma intoetsen, hetgeen dan ook in het volgende hoofdstuk zal gebeuren.

5. Algemene procedure voor het opstellen van een programma

In dit hoofdstuk zullen we ons bezighouden met het opstellen van programma's.
Het voorafgaande programma was in feite een zeer eenvoudig programma. Men zou wellicht kunnen opmerken dat een dergelijke renteberekening ook zo met de hand kan worden uitgevoerd. Anders gezegd: het gebruik van de computer komt hier wel erg overdreven voor. Dat is natuurlijk juist, maar we dienen wel te beseffen dat het programma ook uitgebreid kan worden, zodat meer ingewikkelde berekeningen kunnen worden uitgevoerd. Terzijde merken we op, dat het de computer volledig ontgaat of een programma nu al of niet ingewikkeld is. De computer is en blijft een domme automaat die precies uitvoert wat hem is opgedragen.

Als we een probleem op de computer willen uitvoeren zullen we in het algemeen een aantal fasen doorlopen:
1. omschrijving van de probleemstelling;
2. het zoeken naar een oplossingsmethode;
3. het schrijven van een programma op basis van de gevonden oplossingsmethode;
4. het uittesten van het programma;
5. het documenteren.

Deze fasen zullen we nu aan de hand van een eenvoudig voorbeeld toelichten.

Als voorbeeld nemen we het probleem: het bepalen van de som van de termen van een reeks. De reeks waar het om gaat heeft de getallen 1, 2,3,4 tot en met 100.
Kortom, het eerste getal is 1 en ieder volgend getal krijgen we door 1 bij het vorige getal op te tellen.
Het laatste getal van de reeks is 100.
Hiermede is fase 1 achter de rug: de probleemstelling is immers omschreven.
Fase 2 behelst het vinden van een oplossingsmethode. Soms spreekt men in plaats van de oplossingsmethode van het algorithme. Onder een algorithme verstaan we een aantal stappen die,als ze worden uitgevoerd, uiteindelijk tot het gewenste resultaat leiden. In veel gevallen eindigt het programmeren al in deze situatie. Kortom, er wordt in het geheel geen oplossingsmethode gevonden. Beschouwen we de vraagstelling dan ligt één oplossing wel erg voor de hand nl. tel 1 en 2 op, tel hier 3 bij op en vervolgens 4, enz.
Een programma dat deze handelingen uitvoert zou de volgende gedaante kunnen hebben:

```
10 LET TERM1=1
20 LET TERM2=2
30 LET TERM3=3
  ........
  etc.
1000 LET TERM100=100
1010 LET SOM=(TERM1+TERM2+...etc. +TERM100)
1020 PRINT SOM
```

Deze oplossing riekt naar het zoeken naar eenvoudiger wegen. Bovendien dienen we te beseffen dat de genoemde oplossing erg star is. Als we ooit nog eens een gelijksoortig probleem hebben: bijvoorbeeld we moeten de som van een reeks getallen bepalen waarbij ieder getal verkregen wordt door een constante bij het voorafgaande getal te tellen, dan is de genoemde oplossing volkomen ontoereikend.
Beschouwen we het probleem nader dan valt op dat als we de 1ste en de 100ste en hierna de 2de en 99ste en hierna de 3de en de 98ste term optellen, als som steeds 101 verkregen wordt.
Zo kunnen we alle termen van de reeks in 50 paren verdelen die ieder 101 als som hebben. De totale som kunnen we dus vinden door het aantal paren te vermenigvuldigen met de som van de begin- en de eindterm:

```
AANTAL PAREN*(BEGIN+EIND)
```

Het aantal paren is hier gelijk aan het aantal termen gedeeld door 2. Een totaal aantal wordt vaak met de letter N aangeduid; zo vinden we uiteindelijk:

$$\frac{N}{2} * (BEGIN+EIND)$$

Hiermede eindigt fase 2. De formule lijkt ons een goede oplossingsmethode te geven, een oplossingsmethode die bovendien nog zonder problemen in een BASIC-programma lijkt te kunnen worden omgezet. Fase 3 behelst het omzetten van de oplossingsmethode in een BASIC-programma. Als de oplossingsmethode relatief gezien tamelijk ingewikkeld is zullen we trachten de oplossingsmethode in een aantal min of meer zelfstandige brokken te verdelen. Op een afzonderlijk lijstje omschrijven we de betekenis van de variabelen en vermelden we de naam die iedere variabele in het programma krijgt. Hierna slaan we dan aan het omzetten van de oplossingsmethode in een reeks instructies. De totale reeks instructies vormt dan het uiteindelijke BASIC-programma.
In ons geval was de oplossingsmethode eenvoudig en we volstaan daarom met het geven van het programma:

```
10 LET BEGIN=1
20 LET EIND=100
30 LET N=100
40 LET SOM=N/2*(BEGIN+EIND)
50 PRINT SOM
```

(Antwoord dient 5050 te zijn.)
We merken hierbij op dat haakjes in uitdrukkingen zijn toegestaan en dat deze op dezelfde wijze worden gebruikt als in de 'gewone' algebra. Merk tevens op dat de gehele formule nu op een regel geplaatst is. De 2 staat naast de schuine deelstreep en niet er onder. Dit zou problemen kunnen opleveren met betrekking tot de vraag of nu eerst de vermenigvuldiging 2 * (BEGIN + EIND) zou worden uitgerekend (hetgeen uiteraard niet gewenst is) dan wel dat eerst de deling N/2 wordt uitgevoerd en het resultaat hiervan met de uitdrukking tussen haakjes wordt vermenigvuldigd (dit is wel de bedoeling).
Aan eventuele twijfel zouden we een eind kunnen maken door extra haakjes te plaatsen:

```
40 LET SOM=(N/2)*(BEGIN+EIND)
```

In dit geval hoeft dit echter niet omdat vermenigvuldigen en delen 'gelijke prioriteit' hebben, dat wil zeggen geen van beide heeft bij het uitwerken voorrang en de berekening wordt dan gewoonweg van links naar rechts uitgevoerd. We merken op dat deze regel duidelijk verschilt

met de door ons op school geleerde regel 'Meneer van Dalen wacht op antwoord' waarbij vermenigvuldigen altijd vóór delen gaat.
Bij BASIC geldt dat optellen en aftrekken ook gelijke prioriteit hebben en in een uitdrukking van links naar rechts worden uitgevoerd. Vermenigvuldigen en delen gaan in BASIC wel voor optellen en aftrekken. In een appendix worden nog de uitgebreide regels gegeven, omtrent de vraag hoe uitdrukkingen worden uitgewerkt.
Met de regel voor ogen 'plaats in geval van onzekerheid altijd haakjes' zullen we in de praktijk echter weinig problemen ervaren.

Fase 4 'het testen van het programma', kan ons voor grote problemen plaatsen. Een eerste aanwijzing wordt gegeven als tijdens het uitvoeren van het programma geen onverwachte foutmelding verschijnt. Dit is echter geenszins een garantie voor de juistheid van het programma. Sterker nog, een volledige garantie verkrijgen we nooit.
Als we in plaats van:

```
10 LET BEGIN=1
```

hadden ingetoetst:

```
20 LET BEGIN=2
```

dan zal de computer stellig geen enkele fout melden en als we zelf de fout niet tijdig ontdekken dan zullen we het antwoord gewoonweg als juist overnemen.
In dit geval zouden we het antwoord uiteraard wel met de hand kunnen uitrekenen. Dat is echter flauw omdat het in de praktijk veelal gaat om programma's waar we de antwoorden nu juist niet met de hand kunnen uitrekenen (althans praktisch niet).
Meestal kan men bij het testen de volgende strategieën toepassen:

a. Kijk of er voor het probleem al eens antwoorden gegeven zijn en controleer deze dan aan de hand van uw programma.
b. Probeer met eenvoudige getallen een en ander na te rekenen.
c. Als het een programma betreft dat zeer veel routinematig gebruikt gaat worden wees dan extra voorzichtig en test zoveel mogelijk delen van het programma met de hand uit.
d. Voer extra PRINT-instructies in om tussenresultaten af te drukken. Zo kan men aan de hand van die tussenresultaten het programma als het ware op de voet volgen.

In ons geval zouden we de procedure kunnen testen aan de hand van een reeks die wel dezelfde kenmerken draagt, doch aanmerkelijk eenvoudiger is.
Beschouw bijvoorbeeld eens de reeks:

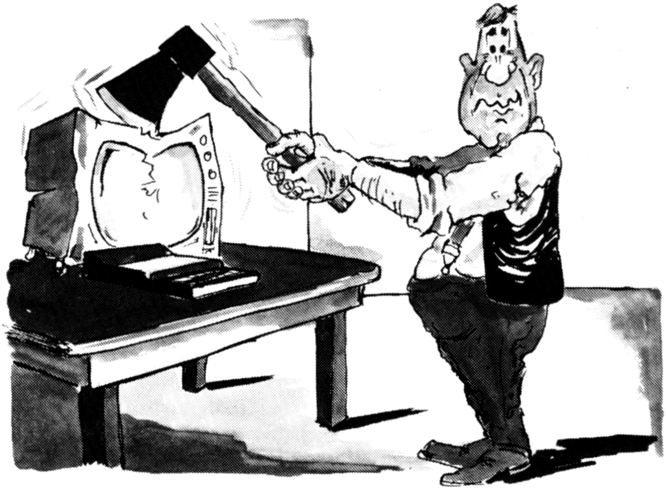
1,2,3 en 4

waar de gevonden formule uiteraard evenzeer van toepassing is (dit kan men trouwens ook uittesten).
Het testprogramma zou dan luiden:

```
10 LET BEGIN=1
20 LET EIND=4
30 LET N=4
40 LET SOM=N/2*(BEGIN+EIND)
50 PRINT SOM
```

De laatste fase, fase 5, wordt doorgaans uit laksheid overgeslagen. Deze luiheid wordt dan naderhand bestraft.
Neem in de documentatie altijd de volgende zaken op:
a. korte omschrijving van de probleemstelling;
b. korte omschrijving van de oplossingsmethode;
c. listing van het programma;
d. omschrijving hoe gegevens moeten worden ingevoerd;
e. voorbeeld van de tekst die verschijnt als het programma wordt uitgevoerd.

6. Herstellen van fouten



In dit gedeelte zal de aandacht worden gevestigd op het herstellen van fouten. De ZX 81 bezit een aantal handige mogelijkheden om fouten te corrigeren en het programma van uitbreidingen te voorzien.
Bovendien controleert de ZX 81 al tijdens het invoeren van iedere regel of er geen overduidelijke fouten worden gemaakt. In dit geval verschijnt bij de plaats des onheils een cursor met de letter S (van Syntax-fout). Deze regel kan niet bij de rest van het programma worden gevoegd, hoe hard we ook op de toets met NEWLINE drukken.
We gaan uit van het volgende reeds bekende programma:

```
10 LET BEGIN=1
20 LET EIND=100
30 LET N=100
40 LET SOM=N/2*(BEGIN+EIND)
50 PRINT SOM
```

Toets dit programma op de reeds beschreven manier in (toets alvorens het programma in te voeren allereerst op NEW om het geheugen schoon te maken). De cursor met het pijltje zal na het intoetsen van het programma naar regel 50 wijzen.
Stel dat we naast de waarde van SOM ook de waarden van BEGIN, EIND en N ter controle willen uitvoeren. Dat kan door regels tussen te voegen.

Toets nu achtereenvolgens in:

```
15 PRINT BEGIN
25 PRINT EIND
35 PRINT N
```

Merk op hoe iedere regel na het indrukken van NEWLINE keurig in het bovenstaande programma wordt gevoegd. Voer het programma nu maar eens ter controle uit met behulp van het commando RUN.

Wil men deze regels weer verwijderen ... ? Geen nood, toets in:

```
15
25
35
```
(N. B. steeds gevolgd door NEWLINE)

Merk op dat de regels nu inderdaad uit het programma zijn verwijderd. Hieruit putten we de algemene wijsheid dat men een willekeurige regel uit een programma kan wissen door het regelnummer van de desbetreffende regel in te typen gevolgd door NEWLINE.

We gaan nu opzettelijk een 'foute' verbetering aanbrengen, nl. door in te typen:

```
20 LET EINT=100
```

Merk op dat de oorspronkelijke regel nu vervangen is door de nieuwe, en de naam EIND is nu vervangen door EINT. Dit biedt trouwens al een krachtige methode voor het herstellen van fouten: type het regelnummer van de desbetreffende regel in, gevolgd door de correcte instructie. Uiteraard is het interessant om te weten of het programma nu inderdaad een onoverkomelijke fout bevat.
We geven het commando:

```
RUN
```
(gevolgd door NEWLINE)

en inderdaad, in plaats van het correcte antwoord verschijnt nu een foutboodschap nl. :

```
2/40
```

Het tweede getal, met andere woorden het getal 40, geeft het regelnummer aan alwaar de fout tijdens het uitvoeren van het programma werd ontdekt en het tweede getal (2) geeft een indicatie omtrent het type fout. In appendix 1 zien we dat foutboodschap 2 betrekking heeft

op een variabele die in een uitdrukking wordt gebruikt zonder dat deze variabele van te voren van een waarde is voorzien.

We kunnen nu weer op NEWLINE drukken dan wel het commando LIST geven. In beide gevallen verschijnt het programma weer (kortom, de listing wordt getoond). Nu bevat regel 40 de variabele EIND die inderdaad nog niet eerder van een waarde is voorzien; de waarde 100 is immers aan de variabele EINT toegekend, en EINT is een andere variabele dan EIND.
In dit geval kunnen we de fout op twee manieren herstellen.
Allereerst kunnen we in regel 40 de naam EIND door EINT vervangen.
Eveneens zouden we in regel 20 EINT door EIND kunnen vervangen.
Uit eerbied voor de Nederlandse taal kiezen we voor de eerste oplossing.
Men zou dit nu kunnen doen door opnieuw in te toetsen:

```
20 LET EIND=100
```

Er is echter een tweede methode die in veel gevallen veel plezieriger verloopt.
Deze methode berust op de cursor-besturing. Beschouw daartoe de pijlen die boven de toetsen 5, 6, 7 en 8 staan. Met de omhoog en omlaag wijzende pijlen kunnen we de cursor met het pijltje in de aangegeven richting verplaatsen. Probeer het maar eens (niet vergeten op SHIFT te drukken terwijl op toets 6 of 7 wordt gedrukt). Op deze wijze kunnen we een bepaalde regel uitzoeken.
Zorg er nu voor dat na bovenstaande oefening de cursor met het pijltje naar regel 20 wijst.
Druk vervolgens in:

EDIT (d.w.z. SHIFT en 1)

en de aangewezen regel verschijnt onderaan de regel om opnieuw bewerkt te worden.
Nu kunnen we de cursor die zich onderaan het beeldscherm bevindt besturen door middel van de pijltjes naar links en naar rechts (bevinden zich boven de toetsen 5 en 8). Probeer dit ook maar eens uit.
Zorg er uiteindelijk voor dat de cursor rechts van het foute symbool staat, met andere woorden we zien wat op de foto op de volgende bladzijde is afgebeeld.
Druk vervolgens op:

RUBOUT (vergeet niet op SHIFT te drukken)

en voer vervolgens de juiste letter in:

D

```
10   LET BEGIN=1
20 > LET EINT=100
30   LET N=100
40   LET SOM=N/2*(BEGIN+EIND)
50   PRINT SOM




20   LET EINT[L]=100
```

Vervolgens toetsen we weer op NEWLINE en het programma is gereed. Merk op dat met RUBOUT een symbool wordt gewist. Men kan een aantal symbolen wissen door meer keren achter elkaar op RUBOUT te drukken (waarbij de toets SHIFT uiteraard ingedrukt wordt gehouden). De getoonde methode dient men zelfs altijd te volgen als tijdens het intoetsen van een regel van een programma een fout wordt gemaakt die onmiddellijk door de computer wordt herkend (syntax-fout).
Toets maar eens in:

```
50 PRINT +SOM
```

Het plus-teken hoort hier niet thuis en dit wordt duidelijk gemaakt door een extra cursor met de letter S (van Syntax-fout) die voor de plaats des onheils zichtbaar wordt. Deze cursor wordt pas zichtbaar na het drukken op NEWLINE. We kunnen in dit geval de zin zelfs niet eerst in het programma opnemen door op NEWLINE te drukken (en dat is in feite maar goed ook!). Het toetsen van NEWLINE heeft behalve het tonen van de cursor geen enkel effect: de ZX 81 eist dat de ontdekte fout eerst wordt hersteld.
De fout herstellen we wederom door de cursor naar de plaats van de fout te sturen. Het plus-teken wordt vervolgens verwijderd door op RUBOUT te drukken en nu kan de regel wel in het programma worden opgenomen door op NEWLINE te drukken.

Ten slotte vermelden we hier nog dat de toetsen ⇧ en ⇩ ook nog gebruikt kunnen worden bij het bestuderen van een listing van een programma. Dit komt speciaal van pas als het programma zo groot is, dat het niet meer als geheel op het beeldscherm kan worden getoond.

Door het intoetsen van ⇩ worden regels met hogere regelnummers zichtbaar en andersom door het intoetsen van ⇧ worden regels met lagere regelnummers zichtbaar.

7. Het gebruik van de computer als calculator

In het voorafgaande beschouwden we de computer vooral als apparaat om er programma's mee uit te kunnen voeren.
De ZX 81 biedt ons echter ook de mogelijkheid om instructies zonder regelnummer op te geven. Deze instructies dragen dan het karakter van commando's.
Op deze wijze is het alsof men op een calculator werkt.
We geven een eerste voorbeeld:

```
PRINT 5.341*6.7
```
(NEWLINE)

De computer antwoordt onmiddellijk met:

```
35.7847
```

We hadden bovenstaande ook wel als programma kunnen uitvoeren. Om het verschil duidelijk te laten uitkomen tonen we dat ook.
Het geheugen wordt eerst met het commando NEW gewist. Hierna toetsen we ons 'micro-programmaatje' in:

```
10 PRINT 5.341*6.7
```

Merk op dat nu eerst een regelnummer is ingevoerd. Het antwoord zal pas verschijnen als het commando RUN wordt gegeven, dus:

```
RUN
```

en inderdaad verschijnt het antwoord weer en wel met onder in het beeld de boodschap 0/10. Iedere maal als we weer het commando RUN geven verschijnt het antwoord opnieuw. Dit is in de vorige situatie ('de computer als calculator') niet mogelijk. Een gegeven commando zoals PRINT 5.341 * 6.7 wordt vergeten als het is uitgevoerd.
Terzijde merken we nog op dat in een PRINT-instructie kennelijk ook een rekenkundige uitdrukking mag staan, er wordt nl. een vermenigvuldiging uitgevoerd.

Het volgende voorbeeld toont dat we ook reeksen 'instructie-commando's' kunnen uitvoeren (voor alle zekerheid: wis het geheugen eerst met behulp van het commando NEW).
We toetsen nu in: (N. B. iedere regel steeds gevolgd door NEWLINE waardoor die regel dan ogenschijnlijk ook weer verloren gaat.)

```
LET A=1

LET B=2

LET C=A+B

PRINT C
```

Onmiddellijk na deze laatste regel verschijnt het antwoord, er zijn immers geen regelnummers voor de 'instructies' geplaatst, zodat iedere regel als commando wordt opgevat.
Toch is de computer niet alles vergeten. Toets maar eens in:

```
PRINT B
```

en als antwoord verschijnt 2.
Van deze mogelijkheid kan men dankbaar gebruik maken als men fouten in een programma aan het opsporen is. Door middel van dergelijke directe PRINT-opdrachten (d.w.z PRINT-opdrachten zonder regelnummer) kunnen we de waarden van verschillende variabelen na het uitvoeren van het programma op eenvoudige wijze achterhalen.
De ZX81 biedt nog een afzonderlijk commando om de waarde die aan iedere variabele is toegekend te wissen, nl. het commando:

```
CLEAR
```

Drukken we hierna PRINT B in, dan verschijnt een foutmelding ten teken dat de waarde van de variabele B onbekend is.
Het commando kan tevens van pas komen bij het opsporen van fouten. Als door wijzigingen in het programma bepaalde instructies niet meer worden uitgevoerd zou men dit feit ondanks het uitvoeren van directe PRINT-opdrachten toch wel eens kunnen ontgaan.
De variabelen zouden nl. van te voren al eens een waarde kunnen hebben gekregen. Door middel van CLEAR kunnen we dit ongemak ver-

mijden: alle waarden van de variabelen worden gewist. Het commando CLEAR wordt ook wel eens gebruikt om weer geheugenruimte vrij te maken.
We besluiten dit gedeelte met de melding dat alle instructies behalve de INPUT-instructie als commando gebruikt kunnen worden.
In de praktijk blijken echter voornamelijk de directe PRINT-instructies van belang te zijn en wel om, zoals gezegd, fouten in programma's op te sporen. Want laten we wel wezen, we kopen geen computer om deze als calculator te gebruiken. Het enige juiste voer is en blijft toch een programma.

8. De LET-instructie

In het voorafgaande werd in elk programma-voorbeeld de LET-instructie gebruikt. Zonder twijfel is de LET-instructie een van de meest gebruikte BASIC-instructies en enige aandacht voor deze instructie is alleen daarom al zonder meer op zijn plaats.

Bij een instructie zoals:

```
30 LET N=100
```

dienen we het teken = niet als een gelijkheid (in de zin van 'N is gelijk aan 100') op te vatten.
We kunnen de instructie het beste vertalen door 'N wordt 100' of 'N krijgt de waarde 100'.
Het is raadzaam om het teken = in een LET-instructie daarom altijd als 'wordt-teken' aan te duiden. Een en ander wordt treffend geïllustreerd aan de hand van de volgende instructie:

```
30 LET N=N+100
```

Volgens de algebra is een uitdrukking als N = N + 100 gewoon onzin (men spreekt dan van een 'strijdige vergelijking'). Volgens BASIC is dit een uitdrukking die aangeeft, dat de nieuwe waarde van N verkregen <u>wordt</u> door bij de oude waarde 100 op te tellen.

Als we hierna een instructie zouden aantreffen als:

```
60 LET N=40
```

dan vervalt de oude waarde weer, deze <u>wordt</u> nl. vervangen door de nieuwe waarde 40.

Na LET volgt altijd de naam van een variabele en na het teken = ziet men een waarde of een uitdrukking die tot een waarde leidt.

Bij veel BASIC-versies is het toegestaan om LET weg te laten. Hier mag men bijvoorbeeld schrijven:

```
40 N=100
```

Bij de ZX 81 is dit verboden. Deze wetenschap kan van pas komen als u reeds bestaande BASIC-programma's in ZX 81-BASIC wilt omzetten. Overal waar u ziet staan:

regelnummer variabele = uitdrukking

dient u LET in te voegen.
Bij deze BASIC-versies bestaat trouwens ook de mogelijkheid om meer dan een instructie op een regel te schrijven, bijvoorbeeld gescheiden door het teken : .
Dit is bij de BASIC-versie van ZX 81 ook niet toegestaan. Men dient deze instructies dan ook op afzonderlijke regels te plaatsen.
Ten slotte zien we bij bepaalde BASIC-programma's dat men om geheugen te sparen geen spaties tussen gereserveerde woorden zoals LET plaatst.
Bij de ZX 81 worden na het intoetsen van zo'n gereserveerd woord automatisch spaties geplaatst. Dat is in feite erg plezierig want deze spaties verhogen de leesbaarheid van het programma.

9. REM, PRINT en INPUT

Om de nu volgende instructies te behandelen brengen we weer het programma, om de som van de termen van een reeks te bepalen, in herinnering.
Het programma luidde:

```
10 LET BEGIN=1
20 LET EIND=100
30 LET N=100
40 LET SOM=N/2*(BEGIN+EIND)
50 PRINT SOM
```

Dit is nauwelijks een 'algemeen' programma te noemen. Het berekent nl. de som van de termen voor maar precies één bepaalde reeks.
De oplossingsmethode die echter is gebruikt geldt voor alle reeksen, waarbij een volgende term wordt verkregen door een constante waarde bij de vorige term op te tellen (rekenkundige reeksen).
Het zou dus plezierig zijn dat de gebruiker op eenvoudige wijze de waarde van de eerste term, de laatste term en het aantal termen kan opgeven, en wel zonder daarbij het programma te wijzigen.
Deze mogelijkheid wordt geboden door de INPUT-instructie. Deze instructie heeft steeds de vorm:

INPUT naam van variabele

Tijdens het uitvoeren van het programma introduceert zo'n INPUT-instructie een tijdelijke onderbreking. De gebruiker dient dan een waarde in te toetsen (gevolgd door NEWLINE) en deze waarde zal aan de aangegeven variabele worden toegekend.
Met INPUT-instructies zal ons programma er dus als volgt uitzien:

```
10 INPUT BEGIN
20 INPUT EIND
30 INPUT N
40 LET SOM=N/2*(BEGIN+EIND)
50 PRINT SOM
```

Voer deze wijzigingen maar eens uit!
Als we het programma vervolgens 'runnen' zien we direct dat het scherm als het ware wordt schoongeveegd en in de linker benedenhoek verschijnt de cursor met de letter L. In dit geval is dit het teken dat we nu zelf een getal moeten invoeren*.
We toetsen bij de eerste keer in:

```
1
```

Onmiddellijk hierna zien we weer de cursor L verschijnen en er moet dus weer een getal worden ingevoerd*, nl. een waarde voor de variabele EIND, bijvoorbeeld:

```
100
```

En ten slotte voeren we de waarde voor N in:

```
100
```

Als antwoord verschijnt nu inderdaad weer de bekende waarde 5050.

We kunnen het programma nu direct gebruiken voor andere waarden, bijv. voor het bepalen van de som van de eerste 50 waarden.
Toets eerst RUN in en voer vervolgens in:

```
1
50
50
```

De computer toont nu 1275 als antwoord.
Het zal duidelijk zijn dat de algemene waarde van het programma aanmerkelijk is toegenomen.

* Het is zelfs mogelijk om in plaats van een getal een uitdrukking, bijvoorbeeld 1 * 4 + 3/7 in te voeren. Deze uitdrukking zal dan eerst tot een getal worden uitgewerkt.

Er is echter nog een opvallend gebrek, dat programmeurs aangeven met de uitdrukking 'het programma is weinig gebruikersvriendelijk'. Met de term gebruikersvriendelijk geven we aan dat een gebruiker het programma vrijwel zonder voorkennis moet kunnen gebruiken.
In dit geval is echter veel voorkennis nodig. We moeten van te voren weten dat er allereerst een getal moet worden ingevoerd dat de waarde van de variabele BEGIN dient voor te stellen enz.
Het zou plezierig zijn als we in het programma zelf boodschappen kunnen opnemen. Nu is het inderdaad mogelijk om door middel van PRINT-instructies tekst af te drukken. Deze tekst dient dan wel tussen de tekens ' (aanhalingstekens) te staan. In het onderstaande is een verbeterde versie van het oorspronkelijke programma te zien. De oorspronkelijke regelnummers zijn voor alle duidelijkheid ongewijzigd.
De toegevoegde regels eindigen met een 5.
U kunt deze wijzigingen zelf uitvoeren.
Voor alle duidelijkheid: een spatie in de tekst die tussen aanhalingstekens wordt geplaatst, verkrijgt men door op SPACE te drukken.

```
 5 PRINT "VOER BEGIN IN"
10 INPUT BEGIN
15 PRINT "VOER EIND IN"
20 INPUT EIND
25 PRINT "VOER AANTAL TERMEN IN"
30 INPUT N
40 LET SOM=N/2*(BEGIN+EIND)
45 PRINT "DE SOM IS"
50 PRINT SOM
```

(N. B. merk op dat regel 25 twee regels op het beeldscherm in beslag neemt.)
Als we het programma starten verschijnt allereerst de tekst:

```
VOER BEGIN IN
```

(merk op dat hier geen aanhalingstekens meer staan)
en hierna toetsen we bijvoorbeeld de waarde 2 in enz.
We kunnen ook verschillende zaken in één PRINT-instructie aangeven.
Zo kunnen we regel 45 en 50 combineren tot één PRINT-instructie:

```
50 PRINT "DE SOM IS ";SOM
```

Als u deze wijziging uitvoert dient u regel 45 uiteraard wel te wissen.
De twee zaken die nu worden afgedrukt zijn: de tekst DE SOM IS en de

waarde van de variabele SOM. Omdat deze twee zaken in de PRINT-instructie zijn gescheiden met behulp van het teken ; worden ze direct achter elkaar geplaatst. Hadden we in plaats van de ; een komma, dat wil zeggen het symbool , gebruikt dan had de computer de waarde van SOM in een volgende kolom geplaatst (de computer deelt bij een komma het beeldscherm automatisch in twee kolommen in).

Bij deze PRINT-instructie worden twee af te drukken zaken aangegeven. In de praktijk kan men meer zaken door middel van de tekens ; en , scheiden.
De tekens ; en , kunnen ook als laatste teken in een PRINT-instructie voorkomen. In dit geval wordt een eventueel volgende af te drukken zaak direct na de reeds afgedrukte tekst geplaatst (bij een ;) dan wel in de volgende kolom geplaatst (bij een ,).
Een PRINT-instructie zonder nadere aanduiding heeft tot effect dat er bij het afdrukken een regel wordt overgeslagen.
Een en ander wordt in het onderstaande geïllustreerd:

```
40 LET SOM=45
50 PRINT "DE SOM",
60 PRINT "IS"
70 PRINT
80 PRINT "HET GETAL ";
90 PRINT SOM
```

Dit geeft als resultaat:

DE SOM	IS	(tekst in 2 kolommen)
		(⇦ regel overslaan)
HET GETAL 45		(45 is direct na tekst geplaatst)

De PRINT-instructie is in BASIC in feite opvallend eenvoudig van structuur.
Kort samengevat geldt:
- er kunnen meer zaken worden afgedrukt en wel gescheiden door de tekens ; en ,
- het effect van ; is dat het volgende direct achter het voorafgaande wordt geplaatst
- het effect van de , is dat het volgende in een volgende kolom wordt geplaatst (zo er nog plaats is, anders komt het aan het begin van een nieuwe regel)
- als een PRINT-instructie met ; wordt afgesloten zal een volgende af te drukken tekst direct achter het voorafgaande worden geplaatst

- als een PRINT-instructie met , wordt afgesloten zal een volgende af te drukken tekst in de volgende kolom worden geplaatst
- tekst wordt in een PRINT-instructie altijd tussen aanhalingstekens geplaatst
- PRINT alleen heeft het effect van een regel overslaan
- in een PRINT-instructie mag ook een uitdrukking waarbij nog gerekend moet worden (zgn. expressie) worden opgenomen. Dit werd in hoofdstuk 7 aan de hand van de directe PRINT-instructies gedemonstreerd.

REM-instructie

Het kan ook handig zijn om in het programma zelf informatie, dat wil zeggen aanwijzingen op te nemen die alleen voor de programmeur van belang zijn. Te denken valt aan aanwijzingen over de oplossingsmethode. Hierbij is het dan uitdrukkelijk de bedoeling dat deze aanwijzingen tijdens het runnen van het programma niet op het beeldscherm worden afgedrukt. Ze dienen dus alleen in de listing van het programma voor te komen.
Deze aanwijzingen kan men aanbrengen door vooraan de regel de specifieke term REM te plaatsen. De term REM komt van Remark, dat zoveel als 'opmerking' betekent.
In het nu volgende ziet men het uiteindelijke programma en wel voorzien van REM-'instructies'. De oorspronkelijke regelnummers zijn nog steeds gehandhaafd.

```
 1 REM PROGRAMMA OM SOM VAN
 2 REM EEN REKENKUNDIGE REEKS
 3 REM TE BEPALEN VOLGENS FORMULE
 4 REM SOM=N/2*(BEGIN+EIND)
 5 PRINT "VOER BEGIN IN"
10 INPUT BEGIN
15 PRINT "VOER EIND IN"
20 INPUT EIND
25 PRINT "VOER AANTAL TERMEN IN"
30 INPUT N
40 LET SOM=N/2*(BEGIN+EIND)
50 PRINT "DE SOM IS ";SOM
```

Indien men slechts 1K RAM-geheugen heeft zal men noodgedwongen spaarzaam met de REM-instructie moeten omgaan. Het geheugen is immers snel vol. Beschikt men over voldoende geheugen dan wordt aangeraden om zo veel mogelijk van de REM-instructie gebruik te maken. In feite is dit een van de beste methoden om uw programma te documenteren. Op deze wijze is immers gegarandeerd dat de documentatie bij het programma niet verloren kan gaan.

10. TAB, AT, CLS en SCROLL

Clear Screen

Na het gedeelte over de PRINT-instructie willen we in dit gedeelte nog enkele zaken behandelen die betrekking hebben op het afbeelden (afdrukken van een of andere grootheid of symbool).
Er werd getoond hoe aan de hand van de komma en een punt-komma in een PRINT-instructie, controle wordt verkregen omtrent de positie waar een af te drukken grootheid wordt geplaatst.
BASIC biedt nog een standaard-uitbreiding in de vorm van de TAB-instructie. Deze instructie heeft hetzelfde effect als de tabulator op een schrijfmachine, vandaar ook de naam TAB.

Een voorbeeld van TAB-instructie ziet men in de volgende uitdrukking:

```
10 PRINT TAB 11;"HOOFDSTUK 1"
```

Het gevolg van deze instructie is dat de tekst HOOFDSTUK 1 pas op de 11de positie van de regel wordt geplaatst. Merk de notatie-vorm op: achter TAB volgt direct de kolomaanduiding*. In dit geval wordt als kolomaanduiding een getal geplaatst. We mogen echter ook een variabele achter de term TAB plaatsen. De waarde van die variabele geeft dan de positie van de kolom aan.
Met behulp van de TAB-instructie kunnen we ook 'puntjes'-lijnen op het beeldscherm tekenen.

---

* Bij de meeste BASIC-versies moet men de kolom-aanduiding na TAB steeds tussen haakjes plaatsen. Dit is bij de ZX81 ook toegestaan.

Het nu volgende programma toont de gang van zaken:

```
10 PRINT TAB 1;"X"
20 PRINT TAB 2;"X"
30 PRINT TAB 3;"X"
40 PRINT TAB 4;"X"
50 PRINT TAB 5;"X"
```

Dit programma geeft een schuine lijn als resultaat:

```
X
 X
  X
   X
    X
```

De mogelijkheid om lijnen te tekenen kunnen we nog verder uitbuiten, nl. door gebruik te maken van de nog te bespreken FOR ... NEXT-instructie.

Ten slotte nog een opmerking over het intoetsen van de term TAB. Als we op het toetsenbord kijken zien we de term TAB onder een toets staan (nl. onder de toets P). Termen die onder een toets staan worden bij de ZX 81 opgevat als functies. We dienen voor het intoetsen van dergelijke functies allereerst op de toets FUNCTION te drukken (d.w.z. op SHIFT/NEWLINE). De cursor toont nu de letter F (van FUNCTIE) ten teken dat bij de volgende toetsindruk de naam van de functie wordt ingelezen. Drukken we dus hierna op toets P dan zien we term TAB op het beeldscherm verschijnen.

> NIET MEER VERGETEN: een term onder een toets kunnen we oproepen door eerst NEWLINE in te drukken waarbij SHIFT ingedrukt gehouden wordt (cursor toont dan letter F van FUNCTION) en vervolgens op de desbetreffende toets te drukken.

De ZX 81 biedt nog een andere zeer krachtige plaats-instructie waarmee we op een willekeurige positie van het scherm iets kunnen plaatsen. We doelen op de AT-instructie.
Na AT volgen steeds twee getallen en wel gescheiden door een komma. Deze getallen geven respectievelijk de regel en de positie op de regel aan, waar de af te drukken zaak moet verschijnen.
Om ervaring op te doen kan men het volgende programma uittesten:

```
10 PRINT "VOER X IN"
20 INPUT X
30 PRINT "VOER Y IN"
40 INPUT Y
50 PRINT AT X,Y;"X"
```

We merken hierbij nog op dat met behulp van AT zelfs bepaalde bestaande tekst overschreven kan worden (probeer 0,0 maar eens).Deze AT-instructie is een krachtig hulpmiddel bij het construeren van allerlei grafische voorstellingen.
Vooral de ontwerper van spelletjes zal van deze instructie veel baat ondervinden. Evenals TAB staat AT onder een toets (nl. onder de toets C). Dit houdt in dat hier eveneens eerst op FUNCTION (d. w. z. SHIFT en NEWLINE) gedrukt moet worden.

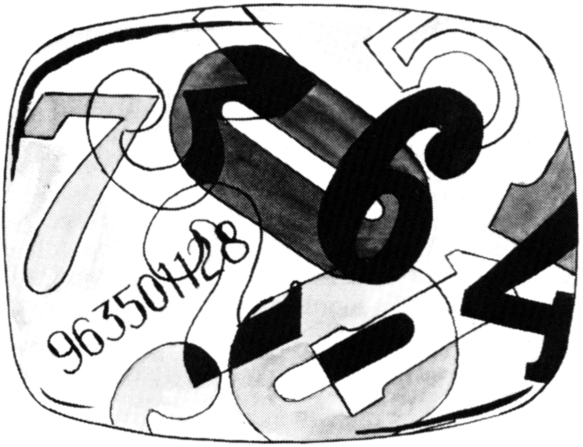
Het kan bijzonder plezierig zijn om het scherm in een keer te wissen. Dat kan, en wel met behulp van de CLS-instructie (CLear Screen-instructie).
Toets maar eens in:

`CLS` (toets V gevolgd door NEWLINE)

en ziedaar, het beeldscherm is gewist.

CLS kan ook als instructie worden opgenomen evenals de laatste instructie die we in dit hoofdstuk zullen behandelen: de SCROLL-instructie. Deze instructie heeft een duidelijk effect als er tekst op het beeldscherm zichtbaar is. We zien nl. dat alle regels een plaats naar boven schuiven, waarbij de bovenste regel uiteraard van het scherm verdwijnt.

11. Over getallen



Ondanks het feit dat computers veel meer kunnen dan alleen maar rekenen, blijft het rekenen een van de zeer belangrijke toepassingsgebieden.
Wie aan rekenen denkt, denkt aan getallen.
In dit hoofdstuk zullen we dieper ingaan op de representatievormen waarin getallen kunnen worden voorgesteld.

Allereerst zijn er de integer-getallen. Dat zijn de gehele getallen 0,1, 2,3, etc. alsmede -1,-2,-3, etc.
De term 'etc.' in voorafgaande zin heeft wel zijn beperkingen, want gezien de beperkte geheugenruimte om gehele getallen op te slaan is er duidelijk een grootste en kleinste geheel getal dat nog net in deze ruimte past.
Voor de ZX81 zijn deze getallen respectievelijk 4294967295 en -4294967296.
Nu zijn dit absoluut gezien wel grote getallen, maar er zijn toch berekeningen waarbij deze getallen niet meer toereikend zijn. Denk maar eens aan de sterrenkunde waar men gewend is aan zeer grote getallen.
Behalve de gehele getallen bestaat er natuurlijk ook de behoefte aan getallen met een decimale punt ('de komma') zodat we bij de deling 1/2 keurig de waarde 0.5 krijgen en niet 0 of 1.
Met deze laatste categorie is het lijstje van verschillende soorten getallen met betrekking tot de ZX81 compleet.
Voor de overzichtelijkheid worden de drie groepen in het onderstaande

nog eenmaal opgesomd:

a. gehele getallen: ... -4, -3, -2, -1, 0, 1, 2, 3, 4 ... Dit zijn de integergetallen.
b. getallen met een decimale punt, bijv. 0.5 en -4.7304. Dit zijn de gebroken getallen. Andere namen zijn reële- of real-getallen.
c. getallen die ook zeer grote (en zeer kleine) waarden kunnen representeren. Dit zijn de getallen die in de zgn. wetenschappelijke notatie worden geschreven.

In het nu volgende zullen we ons eerst bezighouden met de eerste twee categorieën. Aan het eind van dit hoofdstuk komt categorie c aan bod.

Bij BASIC geldt, dat zolang een variabele nog een geheel getal is, deze ook als zodanig wordt opgeslagen. Zodra door een berekening een waarde met een decimale punt ontstaat zal in het geheugen een andere notatie worden gehanteerd. Als gebruiker merken we hier in feite niets van. In een aantal gevallen wil men een getal met een decimale punt weer zo goed mogelijk in een geheel getal omzetten.
BASIC biedt hiertoe de volgende functie:

```
INT
```

Nu hebben we het algemene begrip 'functie' nog niet behandeld. Dit begrip wordt in het volgende hoofdstuk uitvoerig besproken. We volstaan hier met op te merken dat een functie niets meer is dan een voorschrift om een bepaald getal in een ander getal om te zetten (de functie 'kwadraat' is een voorschrift dat het getal 3 in getal 9 omzet; het voorschrift luidt: vermenigvuldig het getal met zichzelf).
Met de uitleg die bij het volgende programma wordt gegeven zullen er wel geen problemen meer ondervonden worden. Het onderstaande programma illustreert de functie INT:

```
10 LET A=2.7
20 LET B=INT(A)
30 LET C=INT(A+0.5)
40 PRINT "INT(2.7)=";B
50 PRINT "INT(2.7+0.5)=";C
```

Dit programma geeft als resultaat:

```
INT(2.7)=2
INT(2.7+0.5)=3
```

In regel 10 wordt aan de variabele A de real-waarde 2.7 toegekend. In regel 20 wordt aan de variabele B de waarde toegekend die we krijgen door de INT-functie op A, met andere woorden het getal 2.7, toe te

passen.
Het resultaat zal dat getal zijn, dat verkregen wordt door het oorspronkelijke getal naar beneden af te ronden.
Zo zal B de integer-waarde 2 krijgen en de PRINT-instructie op regel 40 bevestigt dit. In regel 30 wordt aan C de waarde INT (2.7 + 0.5) toegekend. Door nu bij 2.7 de waarde 0.5 op te tellen, verkrijgt men in feite als resultaat dat 2.7 zal worden afgerond tot het dichtstbijliggende gehele getal, met andere woorden het getal 3.

> ONTHOUDT: een willekeurig getal A kan men afronden door middel van de uitdrukking INT (A+0.5)

Bij negatieve getallen zouden er problemen kunnen ontstaan. Hierbij merken we op dat vooral de INT-functie bij sommige BASIC-versies nogal verschillend wordt geïnterpreteerd. Dit is dan tevens een waarschuwing met betrekking tot het klakkeloos overnemen van BASIC-programma's. Het nu volgende voorbeeld illustreert de gang van zaken bij de ZX81:

```
PRINT INT(-3.1)
```
geeft -4

Het voorbeeld toont nogmaals dat bij de ZX81 de INT-functie naar beneden, dat wil zeggen naar kleinere getallen, afrondt.

Bij sommige rekenproblemen is men niet in de grootte van het getal geïnteresseerd, maar alleen in het teken. Beschouwen we het teken van een getal dan zijn er drie mogelijkheden, nl.:

a. Of het getal is negatief. In dit geval ziet men het teken - voor het getal staan.
b. Of het getal is positief. In dit geval noteert men het teken gewoonlijk niet voor het getal (bij de ZX81 is dit zelfs niet toegestaan), desalniettemin wordt het teken + verondersteld.
c. Het getal heeft geen teken. Alleen aan het getal 0 is deze uitzonderingspositie toebedeeld.

Het teken van een getal, en wel uitgedrukt als de waarde -1, 1 of 0 kan men bepalen aan de hand van de functie:

SGN

Dit wordt in het volgende programma geïllustreerd:

```
10 LET A=-2.7
20 LET B=2.7
30 LET C=0
40 LET AS=SGN(A)
```

```
50 LET BS=SGN(B)
60 LET CS=SGN(C)
70 PRINT "AS=";AS
80 PRINT "BS=";BS
90 PRINT "CS=";CS
```

Dit programma geeft als resultaat:

```
AS=-1
BS=1
CS=0
```

De regels 10,20 en 30, alsmede 70,80 en 90 zullen geen problemen leveren. In de regels 40, 50 en 60 wordt de SGN-functie gebruikt. De resultaten spreken na het voorafgaande overigens voor zichzelf.

Het laatste onderwerp van dit hoofdstuk vormen de zeer grote en zeer kleine getallen.
Beschouw eens het volgende getal:

```
56540000000000000
```

Dit is met recht een astronomisch groot getal ... het zou de afstand tussen een of ander hemellichaam en onze aarde (bijv. in kilometers) kunnen voorstellen.
Een dergelijk getal kunnen we niet zomaar als getal intoetsen. Om dit probleem te ondervangen is een notatie ingevoerd waarbij ieder getal kan worden voorgesteld als:

een getal x macht van 10

Voor het voorbeeld:

5654×10^{13}

Het eerste getal (5654) noemt men de mantisse en het tweede (10^{13}) noemt men de exponent.
Nu dienen getallen altijd keurig op een regel geschreven te worden; het getal 13, dat zoals getoond rechts boven de 10 wordt genoteerd, dient in BASIC dan ook anders geschreven te worden.
De afspraak is eenvoudig: noteer eerst de mantisse, dan de letter E en noteer ten slotte de macht van 10, met andere woorden de exponent.
Dus voor het voorbeeld:

```
5654E13
```

Hetzelfde gebruik hanteert men bij zeer kleine getallen.

Bijvoorbeeld:

```
0.00000000000005654
```

is gelijk aan:

$$5654 \times 10^{-13}$$

en dit wordt dus geschreven als:

```
5654E-13
```

Bedenk dat dit een volledig correcte schrijfwijze van getallen is en dat getallen dus ook volgens deze notatie bij een INPUT-instructie mogen worden ingevoerd.

Om ervaring op te doen raden we ten slotte aan om het volgende programma eens voor verschillende getallen uit te proberen:

```
10 INPUT A
20 INPUT B
30 PRINT A*B
```

12. Functies met betrekking tot getallen

Als we een wetenschappelijke calculator bekijken dan vallen de vele termen op, die op de zgn. wetenschappelijke functies slaan. In feite zijn in die calculator al een aantal standaard-programma's voor de berekening van de aangegeven functies opgenomen. Ook bij de ZX81 zijn standaard-programma's in het ROM-geheugen, voor de berekening van verschillende functies, opgenomen.
Alvorens we de notatie-wijze van functies vermelden, beschouwen we de begrippen functie-waarde en argument.
Een functie is heel in het algemeen een voorschrift (een 'recept') dat aan een getal een ander getal toekent. Zo is het voorschrift bij de functie 'de wortel' als volgt: de wortel van een zekere waarde is dat getal, dat met zichzelf vermenigvuldigd weer die zekere waarde oplevert. Volgens dit voorschrift is de wortel uit 9 de waarde 3, omdat 3 x 3 de waarde 9 oplevert. Het getal dat de functie omzet noemen we het argument en de verkregen waarde wordt functie-waarde genoemd.
In het algemeen geldt in BASIC de volgende notatie met betrekking tot functies:

functie-aanduiding (argument)

Bijvoorbeeld:

```
SQR(9)
```

Hier vormt de term SQR de functie-aanduiding (Engels: SQuare Root) en 9 is het argument van de functie. Met andere woorden $\sqrt{9}$ wordt in BASIC als SQR(9) geschreven. Als we de directe PRINT-instructie geven:

```
PRINT SQR(9)
```

dan wordt inderdaad het getal 3 als functie-waarde gevonden.
Bij de ZX81 mag men de haakjes weglaten, tenminste als de uitdrukking rechts van de functie-aanduiding een enkelvoudig getal is en geen uitdrukking. We nemen dit gebruik echter niet over, omdat het gebruik van haakjes volledig aansluit bij standaard-BASIC. Bovendien worden met het gebruik van haakjes alle mogelijke misverstanden vermeden bij meer ingewikkelde uitdrukkingen.
Bijvoorbeeld:

```
PRINT SQR(2+SQR(16+9)+4/2)
```

zou zonder haakjes stellig tot een ander antwoord leiden.
Aan de hand van dit laatste voorbeeld ziet men trouwens, dat het argument van een functie ook een uitdrukking mag zijn, waarbij nog gerekend moet worden. Er kunnen kennelijk ook weer functies in voorkomen en ook is het toegestaan om variabelen in zo'n uitdrukking op te nemen. De functie-waarde wordt gewoonlijk tot op 8 cijfers nauwkeurig berekend.

Alle termen die functies aangeven zijn bij de ZX81 onder de toetsen geplaatst. Deze termen kunnen we intoetsen door eerst op FUNCTION (d.w.z. op SHIFT en NEWLINE) te drukken. In dat geval toont de cursor de letter F ten teken dat de naam van een functie verwacht wordt.

In het onderstaande zien we een overzicht van de functies, die verder niet meer afzonderlijk zullen worden toegelicht. Deze functies zijn met name voor wetenschappelijke berekeningen van belang en menigeen zal een dergelijke functie nooit gebruiken. Voor deze laatste categorie is deze tabel dan ook weinig interessant. De laatste twee functies van deze tabel werden in het vorige hoofdstuk besproken.

BASIC-aanduiding	algebraïsche notatie	betekenis
SIN(X)	sin(x)	geeft sinus van x; x in radialen
COS(X)	cos(x)	geeft cosinus van x; x in radialen
TAN(X)	tan(x)	geeft tangens van x; x in radialen
ASN(X)	arcsin(x)	geeft boogsinus; antwoord in radialen
ACS(X)	arccos(x)	geeft boogcosinus; antwoord in radialen
ATN(X)	arctan(x)	geeft boogtangens; antwoord in radialen
LN(X)	${}^{e}\ln(x)$	geeft natuurlijke logarithme van x
EXP(X)	e^x	geeft e tot de macht x
SQR(X)	$\sqrt{x}$	geeft wortel uit x
ABS(X)	$\lvert x \rvert$	geeft absolute waarde van x
SGN(X)		geeft -1, 0, 1 afhankelijk van teken van x
INT(X)		rond x altijd naar beneden af

Merk op dat er geen functie is voor het machtsverheffen, terwijl deze functie op de meeste wetenschappelijke calculatoren wel afzonderlijk voorkomt. Bij BASIC verkrijgen we een macht, bijvoorbeeld 2^3 door het bewerkingssymbool * * te gebruiken. Dit symbool kan men met behulp van een enkele toets invoeren, nl. met de toets H (zie plaat op

bladzijde 23).
Voorbeeld van de bewerking machtsverheffen:

```
PRINT 2**3
```

geeft de waarde 2 * 2 * 2 = 8.

We sluiten dit hoofdstuk af met een aantal 'vreemde' functies, en wel in die zin 'vreemd' dat er geen argument is. Met recht kan men dan ook betwijfelen of het nog functies zijn.
Ze worden hier echter als zodanig behandeld omdat de termen die deze 'functies' aangeven onder de toetsen staan. Deze termen kunnen we dus alleen oproepen door eerst op FUNCTION te drukken.

PI (of π) Deze 'functie' geeft direct de waarde π, dat wil zeggen de waarde 3.141592653. De computer bewaart dit getal inderdaad met 10 cijfers, maar er worden er maar 8 weergegeven (N.B. bij de toets M staat het symbool π maar op het beeldscherm ziet men steeds PI.)

RND Deze 'functie' geeft als resultaat een willekeurig getal tussen 0 en 1. De waarde 0 kan eventueel optreden, de waarde 1 echter nooit. Deze functie is met name voor vele spelletjes onmisbaar: het is namelijk de functie die het element van onvoorspelbaarheid introduceert.

De term RND ziet men onder de toets met de letter T staan. Boven deze toets staat de term RAND. Met RAND wordt als het ware de zak met willekeurige getallen (ook wel 'random-getallen' genoemd) geschud, zodat we niet steeds dezelfde 'willekeurige' getallen trekken met behulp van de RND-functie.
We zullen voor het juiste begrip iets meer in detail treden. Er bestaat bij de ZX81 in feite een rij van 65536 willekeurige getallen. Na 65536 trekkingen volgt dezelfde rij weer zodat de getallen in feite toch niet echt willekeurig zijn. Met de RAND-instructie, die steeds de gedaante 'RAND getal' heeft, wordt als het ware de plaats aangegeven waar de RND-functie met het uitlezen van willekeurige getallen uit de gegeven rij zal beginnen.
Bijzondere aandacht vraagt de uitdrukking:

RAND 0 (of dat is identiek: RAND)

In dit geval bepaalt de tijd dat de TV aanstaat de plaats in de rij, waar met uitlezen van de willekeurige getallen begonnen zal worden. Op deze wijze verkrijgt men gewoonlijk werkelijk willekeurige getallen.
Om een en ander tastbaar te maken, moet u de volgende programma's maar eens een aantal malen 'runnen' (N.B. het tweede programma met verschillende, tamelijk lange tussenpozen).

```
10 REM PROGRAMMA VALSE DOBBELSTEEN
20 RAND 1
30 LET UITKOMST=INT((RND*6)+1.)
40 PRINT UITKOMST
```

en

```
10 REM PROGRAMMA DOBBELSTEEN
20 RAND 0
30 LET UITKOMST=INT((RND*6)+1.)
40 PRINT UITKOMST
```

Merk op hoe in regel 30 aan UITKOMST een willekeurig getal tussen 1 en 6 wordt toegekend, en wel door de uitdrukking:

```
INT((RND*6)+1.)
```

Als N een of andere integer-waarde voorstelt kunnen we een willekeurig getal tussen 1 en N (met inbegrip van 1 en N) verkrijgen met behulp van de uitdrukking:

```
INT((RND*N)+1.)
```

Voor toekomstige spelletjesmakers kan dit een interessant gegeven zijn.

13. IF ... THEN, GOTO, STOP, CONT, BREAK

In de vorige programma's hadden we steeds te maken met problemen, waarbij de oplossing werd verkregen door een serie instructies één voor één, van het begin tot het einde uit te voeren.
Stel dat we in een programma een drietal gegevens opgeven die dan betrekking hebben op de lengtes van de drie zijden van een driehoek. Met het beschikbare arsenaal aan instructies is het moeilijk om op grond van de gegevens uit te maken of we nu al of niet met een gelijkbenige driehoek te maken hebben.
Pro memorie: een gelijkbenige driehoek is een driehoek waar tenminste twee zijden aan elkaar gelijk zijn. Het ligt voor de hand om in dit geval vragen te construeren als:

is de eerste zijde gelijk aan de tweede?

en zo niet:

is de eerste zijde gelijk aan de derde?

en zo niet:

is de tweede zijde gelijk aan de derde?

Kortom, er zijn instructies nodig waarmee vragen als bovenstaande uitdrukkingen aangegeven kunnen worden.

BASIC biedt een eenvoudige instructie die deze mogelijkheid biedt, nl. de IF . . . THEN-instructie.

Na de term IF ziet men steeds een uitdrukking die al of niet waar is. Als de uitdrukking waar is, wordt de instructie die na THEN is aangegegeven uitgevoerd. Als de uitdrukking onwaar is dan gaat de computer gewoon met de volgende regel van het programma verder.
Zo zou de instructie kunnen voorkomen:

```
IF ZYDE1=ZYDE2 THEN PRINT "GELYKBENIG"
```

Deze uitdrukking geeft aan:

indien (IF) ZYDE 1 gelijk is aan ZYDE 2 dan (THEN) druk de boodschap GELYKBENIG af

In principe kan men alle instructies na de term THEN plaatsen, maar de mogelijkheden worden pas echt aardig als we na THEN de zgn. GOTO-instructie plaatsen.
De GOTO-instructie is een instructie waarmee een sprong naar een bepaald regelnummer wordt aangegeven. De computer dient het programma bij de aangegeven regel voort te zetten.
De GOTO-instructie heeft steeds de volgende gedaante:

GOTO regelnummer

Bijvoorbeeld:

```
GOTO 40
```

Het nu volgende programma toont zowel de IF ... THEN als de GOTO-instructie. Het programma toont een oplossing voor het probleem dat aan het begin van dit hoofdstuk wordt genoemd. In het programma wordt nagegaan of drie getallen die respectievelijk op de drie zijden van een driehoek slaan, betrekking hebben op een gelijkbenige driehoek.

```
10 REM DRIEHOEK-HERKENNING
20 PRINT "VOER ZYDE 1 IN"
30 INPUT ZYDE1
40 PRINT "VOER ZYDE 2 IN"
50 INPUT ZYDE2
60 PRINT "VOER ZYDE 3 IN"
70 INPUT ZYDE3
80 IF ZYDE1=ZYDE2 THEN GOTO 130
90 IF ZYDE1=ZYDE3 THEN GOTO 130
100 IF ZYDE2=ZYDE3 THEN GOTO 130
110 PRINT "NIET GELYKBENIG"
```

```
120 STOP
130 PRINT "GELYKBENIG"
```

De eerste regel betreft een REM-instructie. Dan volgt een PRINT-instructie waarmee wordt aangegeven dat een getal moet worden ingevoerd. Dit getal heeft dan betrekking op de eerste zijde van de driehoek. Door middel van de INPUT-instructie op regel 30 wordt dit getal aan de variabele ZYDE 1 toegekend. Evenzo wordt het tweede getal aan ZYDE 2 en het derde getal aan ZYDE 3 toegekend.
Nu volgt het 'denkwerk' in het programma.
Regel 80 luidt: IF ZYDE 1=ZYDE 2 THEN GOTO 130, met andere woorden als ZYDE 1 gelijk is aan ZYDE 2 dan wordt een sprong naar regelnummer 130 uitgevoerd. Dit houdt tevens in dat de tussenliggende regels (regel 90 t/m 120) worden overgeslagen.
Nu luidt regel 130: PRINT "GELYKBENIG", kortom, in dit geval volgt de boodschap GELYKBENIG, hetgeen uiteraard correct is, omdat er twee zijden van de driehoek aan elkaar gelijk waren.
Als ZYDE 1 nu niet met ZYDE 2 overeenkomt zal de instructie na THEN niet worden uitgevoerd en de computer gaat dus verder met regel 90.
Hier wordt de test uitgevoerd met betrekking tot ZYDE 1 en ZYDE 3.
De constructie is in feite weer identiek aan de reeds behandelde constructie. Indien ook ZYDE 1 niet met ZYDE 3 overeenkomt, volgt een laatste gelijkvormige test.
Als bij geen van de IF ... THEN-instructies een sprong is gemaakt, kortom in geen van de gevallen komt een zijde met een andere zijde overeen, moeten we wel constateren dat de driehoek niet gelijkbenig is. Deze melding volgt dan ook als resultaat van de PRINT-instructie op regel 110.
Nu oppassen! Als we nu geen maatregelen treffen zou hierna de melding GELYKBENIG volgen! Dit probleem wordt opgelost door op regel 120 de nog niet behandelde STOP-instructie te vermelden.

```
STOP
```

Zoals het woord al aangeeft stopt de computer hier met het uitvoeren van het programma.
Voer het programma nu eens als oefening uit en test verschillende situaties.

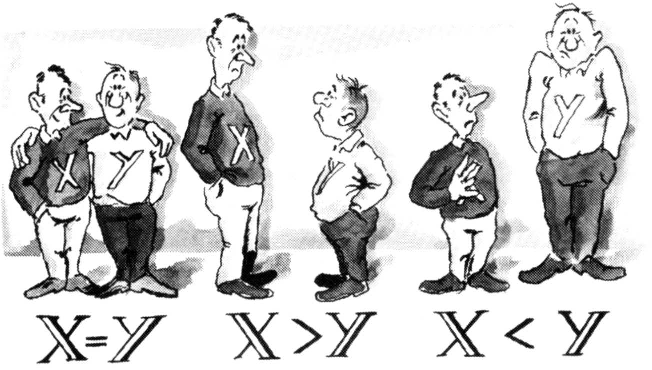
Relatie-tekens

In het programma worden bij de IF ... THEN-instructies twee waarden van variabelen met elkaar vergeleken en wel met het teken = ('gelijk-teken'). Hier wil het teken = inderdaad zeggen 'is gelijk aan'. Men kan ook vragen construeren als 'is ZYDE 1 groter dan ZYDE 2' of 'is ZYDE 1 kleiner of gelijk aan ZYDE 2'. In dit geval worden respectievelijk de tekens > en < = gebruikt. Dergelijke tekens noemt men

relatie-tekens: ze brengen twee grootheden met elkaar in relatie. In het nu volgende zien we een overzicht van de toegestane relatie-tekens.

teken	betekenis	voorbeeld
=	is gelijk aan	IF A=B THEN . . .
<	kleiner dan	IF A<B THEN . . .
<=	kleiner of gelijk	IF A<=B THEN . . .
>	groter dan	IF A>B THEN . . .
>=	groter of gelijk	IF A>=B THEN . . .
< >	ongelijk	IF A<>B THEN . . .

We merken hierbij op dat al deze tekens (dus ook de tekens die uit meer symbolen bestaan) door middel van één toetsindruk moeten worden ingevoerd (zie ook de figuur op bladzijde 23).



AND, OR en NOT

Beschouw nogmaals programma "DRIEHOEK HERKENNING". Hier staan drie nogal gelijkvormige IF-instructies onder elkaar. In feite wordt getest; of ZYDE 1 = ZYDE 2 óf ZYDE 1 = ZYDE 3 óf ZYDE 2 = ZYDE 3 en als tenminste een van deze zaken opgaat springen we naar de instructie PRINT "GELYKBENIG".
BASIC biedt ons de mogelijkheid om dit beknopter weer te geven en wel met behulp van de term OR.
De uitdrukking:

```
ZYDE1=ZYDE2 OR ZYDE1=ZYDE3
```

is waar als tenminste een van beide 'deel-uitdrukkingen' opgaat. Men mag trouwens nog meer 'deel-uitdrukkingen' combineren, bijvoorbeeld:

```
ZYDE1=ZYDE2 OR ZYDE1=ZYDE3 OR ZYDE2=ZYDE3
```

Met behulp van deze uitdrukking zou het programma DRIEHOEK HERKENNING inderdaad aanmerkelijk korter worden.

Naast de term OR kan men de term AND gebruiken. Deze verbindt twee uitdrukkingen en de samengestelde uitdrukking is alleen waar als beide deel-uitdrukkingen waar zijn.
Zo zal bij de instructie:

```
IF A=1 AND B=2 THEN PRINT "A IS 1 EN B IS 2"
```

de PRINT-instructie alleen worden uitgevoerd als zowel A gelijk is aan 1 en B gelijk is aan 2. In alle andere gevallen wordt de PRINT-instructie niet uitgevoerd.

Men mag de termen AND en OR ook combineren en men mag bovendien haakjes gebruiken, bijvoorbeeld:

```
(A=1 AND B=2) OR (A=2 AND B=3)
```

In het nu volgende zien we de betekenis van de termen AND en OR nogmaals in schema aangegeven.
De termen X en Y geven in dit schema deel-uitdrukkingen aan, die waar of onwaar kunnen zijn.

X	Y	X AND Y	X OR Y
waar	waar	waar	waar
waar	onwaar	onwaar	waar
onwaar	waar	onwaar	waar
onwaar	onwaar	onwaar	onwaar

Een deel-uitspraak kan, door er de term NOT voor te plaatsen van waarde verwisselen. In schema:

X	NOT X
waar	onwaar
onwaar	waar

Aan het slot van dit gedeelte nog een spelletje. Bij dit spelletje staat u op een toren en u gooit van deze toren voorwerpen naar beneden. De

natuurkunde laat zien dat voor de valsnelheid geldt:
$V = \sqrt{2 \times 9.8 \times H}$ waarin H de hoogte van de toren voorstelt.

Ga het programma als oefening zelf na:

```
10 REM PISA RAADSPELLETJE
20 PRINT "VOER HOOGTE TOREN IN"
30 INPUT H
40 LET V=SQR(2*9.8*H)
50 PRINT "WAT IS DE VALSNELHEID?"
60 INPUT VSCHAT
70 LET VERSCHIL=ABS(V-VSCHAT)
80 IF VERSCHIL<1 THEN GOTO 110
90 PRINT "U ZIT ER NAAST...."
100 GOTO 50
110 PRINT "GERADEN "
```

Doorgaan of onderbreken

De laatste onderwerpen die we in dit hoofdstuk zullen bespreken zijn de CONT- en de BREAK-commando's.
Met het CONT-commando kunnen we een programma weer voortzetten

als het, bijvoorbeeld door een STOP-instructie, is onderbroken.
Het volgende programmaatje behoeft nauwelijks enige toelichting:

```
10 PRINT "A"
20 STOP
30 PRINT "B"
40 STOP
50 PRINT "C"
```

Hierbij vinden we het volgende resultaat:

```
RUN     (gevolgd door NEWLINE om programma te starten)
A       (hierna wordt programma onderbroken)
CONT
B       (hierna wordt programma weer onderbroken)
CONT
C
```

Als tijdens het uitvoeren van een programma het beeldscherm vol geraakt, zal de computer het programma onderbreken en foutmelding 5 tonen. Met CONT kunnen we het programma dan weer voortzetten. Uiteraard wordt dan wel eerst (automatisch) het beeldscherm weer gewist.

ONTHOUDT: beeldscherm vol . . . gebruik CONT

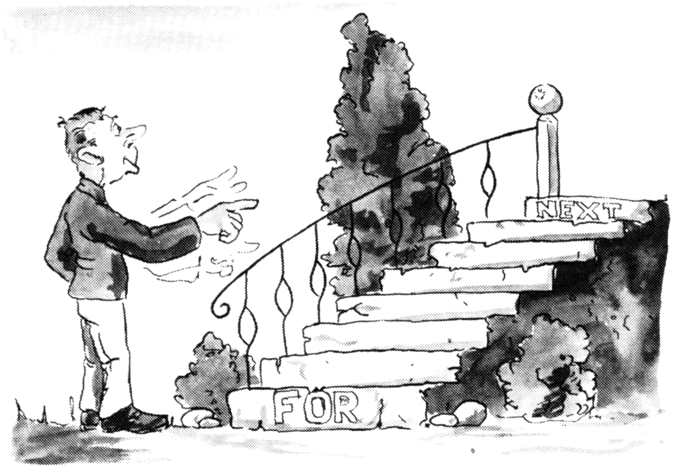
Met het BREAK-commando kunnen we in feite 'met brute kracht' het uitvoeren van een programma onderbreken.
Beschouw bijvoorbeeld eens het volgende illustratie-programmaatje:

```
10 GOTO 20
20 GOTO 10
```

Starten we het programma, dan zal de computer eerst naar regel 20 springen, en aldaar aangekomen zal weer naar regel 10 gesprongen worden en vandaar weer naar regel 20 enz.
We spreken in dit geval van een 'eindeloze lus'. Uiteraard zou men nu de stekker uit het stopcontact kunnen trekken, maar dan raakt het programma ook verloren. Nu is dit wel niet zo'n imposant programma, maar in de praktijk treedt zo'n lus ook wel eens op in een relatief groot programma.
In dit geval kan men het BREAK-commando geven en op het beeldscherm ziet men dan de foutmelding D, alsmede het regelnummer

waar de BREAK-actie plaatsvond, verschijnen.
Dit commando komt ook van pas bij het werken met de cassetterecorder: als de knoppen niet goed zijn ingesteld kunnen we met behulp van BREAK ingrijpen (zie ook hoofdstuk 19).

14. FOR ... NEXT-instructie



Iedereen heeft wel eens een gemiddelde bepaald; een gemiddeld rapportcijfer, een gemiddelde afstand, een gemiddeld benzineverbruik enz.
Het gemiddelde van een reeks getallen bepalen we door alle getallen bij elkaar op te tellen, waarna de som gedeeld wordt door het aantal getallen.
Stel nu eens dat we het gemiddelde van bijvoorbeeld 5 getallen met behulp van de computer zouden willen bepalen.
Dit zou aan de hand van het volgende programma kunnen gebeuren:

```
10 LET SOM=0
20 INPUT GETAL1
30 LET SOM=SOM+GETAL1
40 INPUT GETAL2
50 LET SOM=SOM+GETAL2
60 INPUT GETAL3
70 LET SOM=SOM+GETAL3
80 INPUT GETAL4
90 LET SOM=SOM+GETAL4
```

```
100 INPUT GETAL5
110 LET SOM=SOM+GETAL5
120 LET GEMIDDELDE=SOM/5
130 PRINT GEMIDDELDE
```

Dit is overduidelijk een slecht programma: het is star want het is immers maar voor een reeks van één bepaalde lengte geschikt en het is bovendien relatief gezien erg lang. Als we op deze manier het gemiddelde van 100 getallen zouden moeten bepalen ...
Met behulp van de IF ... THEN-instructie kunnen we in feite alle problemen de baas. In het onderstaande volgt een versie van het programma die zeer gebruikersvriendelijk is:

```
10 REM PROGRAMMA GEMIDDELDE
20 PRINT "VOER HET AANTAL IN"
30 INPUT AANTAL
40 LET N=1
50 LET SOM=0
60 PRINT "VOER GETAL NR";N;" IN"
70 INPUT GETAL
80 LET SOM=SOM+GETAL
90 LET N=N+1
100 IF N<=AANTAL THEN GOTO 60
110 LET GEMIDDELDE=SOM/AANTAL
120 PRINT "GEMIDDELDE=";GEMIDDELDE
```

Regel 10,20 en 30 geven geen problemen. Aan AANTAL wordt het aantal getallen toegekend, waarvan het gemiddelde bepaald moet worden. Op deze wijze is het programma geschikt voor reeksen van willekeurige lengtes. Dan worden in regel 40 en 50 twee beginwaarden gegeven nl. N krijgt de beginwaarde 1 en SOM de beginwaarde 0. Met N zullen we het aantal ingelezen waarden bijhouden en bij SOM zullen we steeds ieder ingelezen getal optellen. Op deze wijze zal SOM uiteindelijk de som van alle ingelezen waarden bevatten. Het geven van dergelijke beginwaarden komt in het programmeren zeer veel voor, dus neem de constructie goed in u op. De regels 60,70,80, 90 en 100 vormen als het ware een afgerond geheel, dat zolang N nog maar kleiner of gelijk is aan AANTAL, steeds opnieuw zal worden uitgevoerd. Een dergelijk afgerond geheel wordt ook wel eens met de naam 'blok' aangeduid. De IF-instructie op regel 100 controleert of alle getallen nog niet zijn

ingevoerd. Zolang dat nog niet het geval is wordt teruggekeerd naar regel 60, zodat het blok inderdaad nogmaals zal worden uitgevoerd. Als alle getallen zijn ingevoerd wordt in regel 110 het gemiddelde berekend, en wel door SOM door het aantal getallen AANTAL te delen. Dit resultaat zien we ten slotte als resultaat van de PRINT-instructie. Probeer het programma maar eens uit voor de getallen 12, 10, 9, 13 en 11. Het resultaat dient 11 te zijn.

In dit geval werd een reeks instructies herhaalde malen uitgevoerd, terwijl het aantal door een 'tel-variabele' (de variabele N) werd bijgehouden. N doorliep achtereenvolgens de waarden 1,2,3 ... AANTAL. Een dergelijke constructie komt zo vaak voor, dat BASIC hier een bijzonder handige instructie voor biedt, nl. de FOR ... NEXT-instructie. Deze instructie begint steeds met:

FOR naam van variabele = beginwaarde TO eindwaarde

Bijvoorbeeld:

```
FOR N=1 TO AANTAL
```

Hierna volgen de instructies die voor alle waarden van de 'tel-variabele' (in bovenstaande situatie, de variabele N) moeten worden uitgevoerd. Uiteraard moet worden aangegeven tot welke instructie het blok, met andere woorden de reeks instructies die steeds moeten worden uitgevoerd, zich strekt. De afspraak is dat na de laatste instructie van het blok de 'vlag'

NEXT naam van variabele

wordt geplaatst, bijvoorbeeld:

```
NEXT N
```

We illustreren dit weer aan het programma GEMIDDELDE dat nu met de FOR ... NEXT-instructie wordt opgebouwd.

```
10 REM PROGRAMMA GEMIDDELDE
20 PRINT "VOER HET AANTAL IN"
30 INPUT AANTAL
40 LET SOM=0
50 FOR N=1 TO AANTAL
60 PRINT "VOER GETAL NR";N;" IN"
70 INPUT GETAL
80 LET SOM=SOM+GETAL
90 NEXT N
```

```
100 LET GEMIDDELDE=SOM/AANTAL
110 PRINT "GEMIDDELDE=";GEMIDDELDE
```

Merk op hoe gemakkelijk deze versie zich laat lezen. (N. B. de term TO in een FOR . . . NEXT-instructie dient met de hiervoor aanwezige toets te worden ingevoerd en niet met de toetsen T en O.)

De FOR . . . NEXT-instructie kan nog worden uitgebreid door expliciet aan te geven met wat voor stappen de tel-variabele die na FOR wordt genoemd, iedere keer moet worden opgehoogd (of verlaagd als de aangegeven stapgrootte negatief is).
Deze uitbreiding kunnen we het beste aan de hand van een klein programma illustreren.

```
10 REM DEMONSTRATIE VAN STEP
20 FOR X=0.3 TO 6.8 STEP 0.3
30 PRINT X
40 NEXT X
50 PRINT "EINDE DEMONSTRATIE"
```

Merk de uitbreiding van de FOR . . . NEXT-instructie met "STEP 0. 3" op. In dit geval zal X de waarden 0. 3, 0. 6, 0. 9 enz. aannemen. De laatste waarde zal 6. 6 zijn, omdat hierna de waarde 6. 6 + 0. 3 = 6. 9 volgt en deze waarde is groter dan de aangegeven eindwaarde 6. 8.
Als het gedeelte "STEP etc. " wordt weggelaten, wordt zoals het programma GEMIDDELDE toonde, in feite uitgegaan van STEP1. (N.B. ook STEP dient met de daartoe aanwezige toets te worden ingevoerd, en niet met de toetsen S, T, E en P.)

Nog een aantal opmerkingen naar aanleiding van de FOR . . . NEXT-instructie:
- De variabele na FOR noemt men gewoonlijk de FOR- of telvariabele. Bij de ZX 81 dient de naam van deze variabele uit slechts een enkele letter te bestaan!
- De beginwaarde, eindwaarde en stapgrootte mogen ook door middel van uitdrukkingen worden aangegeven.
- Men mag nooit van buiten door middel van een GOTO-instructie <u>in</u> het blok van een FOR . . . NEXT-instructie springen.
- Een FOR . . . NEXT-instructie mag zelf weer een FOR . . . NEXT-instructie bevatten. FOR. . .NEXT-instructies mogen elkaar nooit gedeeltelijk overlappen. Het volgende illustreert de situatie:

FOR K = 1 TO 20	FOR K = 1 TO 20
. . .	. . .
FOR J = 1 TO 10	FOR J = 1 TO 10
. . .	. . .
. . .	. . .
NEXT J	NEXT K
. . .	. . .
NEXT K	NEXT J
correcte constructie	incorrecte constructie

Aan het slot van dit hoofdstuk tonen we een programma om een functie op het beeldscherm af te beelden ('te plotten'). In dit geval gaat het om de sinus-functie.
Ga als oefening maar eens na hoe een en ander in zijn werk gaat. Om te helpen zijn enkele REM-instructies in het programma opgenomen*:

```
10 REM PLOT-PROGRAMMA
20 FOR K=0 TO 2*PI STEP 2*PI/20
30 LET W=SIN(K)
40 REM NU OPSCHUIVEN EN VERGROTEN
50 LET POS=(W+1)*10+1
60 REM NU AFRONDEN
70 LET POS=INT(POS+0.5)
80 PRINT TAB POS;"*"
90 NEXT K
```

(N. B. bedenk bij regel 30,40 en 50 dat een sinus altijd waarden tussen -1 en 1 heeft.)

* De uitdrukking PI dient met de toets waarbij het teken π staat te worden uitgevoerd.

15. Arrays

Reeksen

We gaan wederom van een reeks getallen uit en stellen ons nu voor om aan de hand van een programma de vraag op te lossen of er tenminste twee getallen aan elkaar gelijk zijn. Nu zou het kunnen zijn, dat toevallig het eerste en laatste getal aan elkaar gelijk zijn, hetgeen inhoudt dat we het eerste getal nu tot en met het invoeren van het laatste getal moeten opslaan. Hetzelfde geldt trouwens voor het tweede getal en het derde enz. Moraal: we moeten alle getallen afzonderlijk opslaan.
Zou men dan toch die onaangename constructie:

```
10 INPUT GETAL1
20 INPUT GETAL2
30 INPUT GETAL3
     etc.
```

moeten gebruiken.
Deze constructie is stellig onhandelbaar als het gaat om een relatief groot aantal getallen.
BASIC biedt voor dit probleem gelukkig een zeer elegante oplossing in de vorm van de array.
Bij een array worden er door middel van een enkele instructie een willekeurig groot aantal variabelen geïntroduceerd (tenminste tot zover het geheugen dat toelaat). Al die variabelen mag men dan als 'gewone'

variabelen behandelen.
In het nu volgende zullen we eerst in het algemeen de array behandelen. Hierna wordt de oplossing van het zojuist genoemde probleem gegeven.

De instructie om de variabelen te introduceren is de DIM-instructie. Deze heeft bij de ZX81 steeds de volgende gedaante:

DIM letter (aantal)

Bijvoorbeeld:

```
DIM A(20)
```

In dit geval mag men de variabelen met de volgende namen gebruiken:

```
A(1), A(2), A(3) ...  A(20)
```

Al deze variabelen, kortom A(1) t/m A(20) vormen dan de array (Nederlands: reeks).
Deze variabelen mag men, zoals gezegd, zoals iedere andere variabele gebruiken. Zo zijn alle volgende uitdrukkingen toegestaan:

```
LET SOM=A(1)+24
LET A(2)=A(2)+SQR(A(4))
LET  A(4)=A(3)**2+A(19)
```

De letter (in dit geval A) noemt men de hoofdnaam en het getal tussen haakjes is de index.
Men mag de index ook indirect door middel van een variabele, ja zelfs door middel van een uitdrukking aangeven.
Zo zijn er dus drie situaties:

A(3)	: index wordt direct door getal aangegeven
A(K)	: index wordt indirect door variabele aangegeven
A(K + 4)	: index wordt indirect door een uitdrukking aangegeven

Het volgende programma geeft een illustratie:

```
10 DIM X(10)
20 FOR K=1 TO 10
30 LET X(K)=100+K
40 PRINT K,X(K)
50 NEXT K
60 LET SOM=X(1)+X(10)
```

```
70 PRINT "SOM=";SOM
```

Dit programma geeft als resultaat:

```
1          101
2          102
3          103
4          104
5          105
6          106
7          107
8          108
9          109
10         110
SOM=211
```

In regel 10 wordt door middel van de DIM-instructie ruimte voor 10 variabelen gereserveerd. De hoofdnaam mag bij de ZX 81 maar met één letter worden aangegeven. In dit geval is dit de letter X. Op deze wijze worden de variabelen X(1), X(2) ... X(10) geïntroduceerd. Regel 20 geeft het begin van een FOR ... NEXT-instructie aan. In regel 30 wordt aan de variabele X(K) de waarde 100 + K toegekend. Zo zal de eerste keer (K = 1) de variabele X(1) de waarde 100 + 1 krijgen. Vervolgens wordt de waarde van X(K) afgedrukt. Zo zal het blok (regel 30 en 40) 10 maal worden doorlopen en zullen aan de variabelen X(1)t/m X(10) op de aangegeven wijze waarden worden toegekend. Aan de hand van de PRINT-instructie kunt u zich hiervan zelf overtuigen.
Na de FOR ... NEXT-instructie volgt de instructie:

```
60 LET SOM=X(1)+X(10)
```

De daarop volgende PRINT-instructie demonstreert dat de computer de beide waarden X(1) en X(10) inderdaad heeft onthouden.

Nu dan het oorspronkelijke probleem: herkennen of in een rij getallen tenminste twee getallen aan elkaar gelijk zijn.
We gaan er van uit dat een rij getallen wordt ingevoerd en wel met behulp van een FOR ... NEXT-instructie. Deze constructie kwamen we in het vorige hoofdstuk tegen:

```
10 PRINT "VOER AANTAL IN"
20 INPUT AANTAL
30 DIM A(AANTAL)
```

```
40 FOR K=1 TO AANTAL
50 PRINT "VOER GETAL NR";K;" IN"
60 INPUT A(K)
70 NEXT K
```

Dit gedeelte zal geen problemen meer opleveren. Merk op hoe in regel 30 ruimte voor de variabelen A(1), A(2) t/m A(AANTAL) wordt gereserveerd. Door middel van de daarop volgende FOR ... NEXT-instructie wordt aan iedere variabele een waarde toegekend.
Hoe kunnen we nu ontdekken of er tenminste twee getallen aan elkaar gelijk zijn?
We zouden kunnen beginnen met het eerste getal, met andere woorden A(1), om dit getal dan vervolgens te vergelijken met A(2), A(3) t/m A(AANTAL). Als bij dit vergelijken twee getallen aan elkaar gelijk blijken te zijn, kunnen we een sprong maken naar een PRINT-instructie die de boodschap TWEE GETALLEN ZYN GELYK afdrukt.
Als deze situatie niet optreedt kunnen we het eerste getal wel vergeten en we herhalen de gehele procedure met het tweede getal. Mocht ook dit geen succes opleveren dan vergeten we tevens het tweede getal en vervolgen de procedure met het derde getal enz.
Bovenstaande werkwijze kan men bondig met twee FOR ... NEXT-instructies aangeven, nl.:

```
80 FOR K=1 TO AANTAL
90 FOR J=K+1 TO AANTAL
100 IF A(K)=A(J) THEN GOTO 150
110 NEXT J
120 NEXT K
130 PRINT "ER ZYN GEEN TWEE GETALLEN GELYK"
140 STOP
150 PRINT "TWEE GETALLEN ZYN GELYK"
```

De kern van de oplossing wordt hier door de regels 80,90 en 100 gevormd. Iedere keer zal A(K) met de waarden A(K + 1), A(K +2) enz. worden vergeleken. Als er een gelijkheid wordt gevonden, vindt de sprong naar de PRINT-instructie op regel 150 plaats. Zo niet, dan wordt eerst instructie 130 uitgevoerd en hierna stopt het programma bij regel 140. Hiermede is de volledige oplossing van ons probleem gegeven.

Meer-dimensionale arrays

We kunnen na de hoofdnaam in een DIM-instructie ook meer cijfers,

en wel gescheiden door komma's, plaatsen. In dit geval spreekt men van meer-dimensionale arrays. Plaatst men 2 cijfers dan gaat het om een 2-dimensionale array, plaatst men 3 cijfers dan gaat het om een 3-dimensionale array enz.
In het nu volgende geven we alleen een demonstratie van een 2-dimensionale array.
Beschouw als voorbeeld de instructie:

```
DIM A(3,4)
```

Hiermee worden de volgende variabelen geïntroduceerd:

```
A(1,1)   A(1,2)   A(1,3)   A(1,4)
A(2,1)   A(2,2)   A(2,3)   A(2,4)
A(3,1)   A(3,2)   A(3,3)   A(3,4)
```

Deze variabelen kan men weer op dezelfde wijze behandelen als de variabelen bij de tot nu toe behandelde '1'-dimensionale array.
Bijvoorbeeld:

```
10 DIM A(3,4)
20 FOR K=1 TO 3
30 FOR J=1 TO 4
40 LET A(K,J)=10*K+J
50 PRINT A(K,J)
60 NEXT J
70 NEXT K
```

We laten het aan de lezer om dit programma zelf uit te testen.
Ten slotte vermelden we nog dat de hoofdnaam van een array gelijk mag zijn aan de naam van een 'gewone' variabele. Zo mogen de variabelen A(1) t/m A(10) naast de variabele A in hetzelfde programma voorkomen. Om onderscheid te maken tussen een 'gewone' variabele en een variabele van een array (ook wel element van een array genoemd) spreekt men bij de laatste van een 'geïndiceerde variabele'.

16. Subroutines

Een subroutine is zo iets als een refrein in een liedje.

Het schaakspel is stellig een van de meest uitdagende spelen voor bepaalde programmeurs. Ieder jaar worden er toernooien gehouden. Het aantal schaakcomputers ... of beter, het aantal schaakprogramma's groeit met de dag.
Bij het schaken wensen we na iedere zet het bord weer te geven. Het is dan plezierig dat we dit uittekenen van het bord in een afzonderlijk programma kunnen beschrijven, waarbij we dan iedere keer als het bord moet worden weergegeven van het 'hoofdprogramma' naar dit deelprogramma kunnen springen. Als het deelprogramma is afgewerkt moet weer naar het 'hoofdprogramma' gesprongen worden alsof er niets gebeurd is.
Gewoonlijk noemen we het afzonderlijke programma waar tijdelijk naar toe wordt gesprongen de subroutine. Het programma van waaruit naar deze subroutine wordt gesprongen zullen we inderdaad het hoofdprogramma noemen. BASIC biedt voor een dergelijke constructie de GOSUB-instructie. Deze instructie heeft steeds de volgende vorm:

GOSUB regelnummer

Het gevolg hiervan is dat naar het aangegeven regelnummer wordt gesprongen alwaar het programma wordt voortgezet. Als zodanig is er geen verschil met de GOTO-instructie. Dat verschil is er echter wel degelijk omdat de computer, eenmaal aangekomen bij de term RE-

TURN, weer terugkeert naar het hoofdprogramma en wel na de regel die volgt op de regel waar de GOSUB-instructie stond.

In het onderstaande maken we de procedure aan de hand van een schema duidelijk.

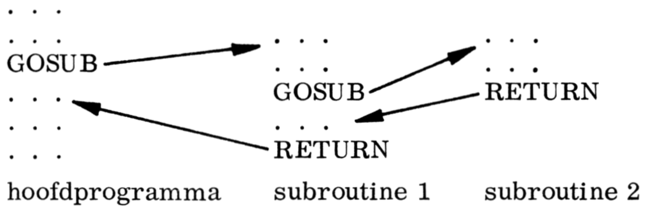
```
10 ...                              ┐
20 ...                              │
   ...                              │
   ...                              │
100 GOSUB 300                       │
110 ...                             │  hoofdprogramma
    ...                             │
    ...                             │
200 GOSUB 300                       │
210 ...                             │
    ...                             │
    ...                             │
270 STOP                            ┘
300 REM BEGIN SUBROUTINE            ┐
    ...                             │  subroutine
    ...                             │
350 RETURN                          ┘
```

In het voorbeeld beslaat het hoofdprogramma regel 10 t/m regel 270. Op regel 100 wordt voor de eerste maal de subroutine aangeroepen.De computer vervolgt het programma met regel 300 enz. Eenmaal bij regel 350 aangekomen keert de computer terug naar het hoofdprogramma en wel naar regel 110. Bij regel 200 aangekomen wordt weer naar regel 300 gesprongen. Na het bereiken van regel 350 volgt weer een sprong terug, maar nu naar regel 210.
Merk de STOP-instructie bij regel 270 op. Op deze wijze wordt vermeden, dat de computer ongewild weer met de instructies van de subroutine zou beginnen. Dit zou trouwens bij regel 350 tot een foutmelding leiden (nl. foutmelding nr. 7: er wordt een RETURN aangetroffen zonder corresponderende GOSUB-instructie).

Het volgende toont een misschien wat merkwaardig voorbeeld, waar de GOSUB- instructie in ieder geval treffend uitkomt.
We laten het aan de lezer zelf over, om gewenste proza in te vullen.

```
10 REM ZELF IN TE VULLEN LIEDJE
20 PRINT "      }
30 PRINT "      }  eerste couplet
40 PRINT "      }  (zelf invullen)
50 GOSUB 150
60 PRINT "      }
70 PRINT "      }  tweede couplet
80 PRINT "      }  (zelf invullen)
90 GOSUB 150
100 PRINT "     }
110 PRINT "     }  derde couplet
120 PRINT "     }  (zelf invullen)
130 GOSUB 150
140 STOP
150 REM REFREIN
160 PRINT "     }
170 PRINT "     }  refrein
180 PRINT "     }  (zelf invullen)
190 RETURN
```

(Wellicht iets voor 'de ballade van de ZX 81').
Ten slotte merken we op dat vanuit een subroutine weer naar een andere subroutine gesprongen kan worden.
Na aankomst van RETURN zal dan worden teruggesprongen naar de subroutine van waaruit de tweede subroutine werd aangeroepen.
Komen we nogmaals RETURN tegen dan keren we weer naar het hoofdprogramma terug.
Als we de deelprogramma's (subroutines) naast elkaar plaatsen wordt het volgende schema gevonden:



Op deze wijze kunnen we veel subroutines aan elkaar rijgen.
De zaken worden erg interessant als we een subroutine zichzelf laten aanroepen. We krijgen dan recursieve subroutines: een onderwerp voor de hogere programmeerkunde dat we dan ook als zodanig niet meer zullen uitspitten.

17. Over letters en andere tekens

Wat is het leven zonder letters. Geen boeken, geen kranten, geen tijdschriften, geen gebruiksaanwijzing voor de ZX81. Het hoeft ons dus niet te verbazen dat computers ook met letters en meer in het algemeen met schrifttekens kunnen manipuleren.
Tekstverwerking vormt heden ten dage een van de belangrijkste toepassingen van de computer. Ook met de ZX81 kunnen we met schrifttekens allerlei handelingen uitvoeren.
Allereerst vermelden we dat het mogelijk is om aan variabelen ook reeksen letters als 'waarde' toe te kennen. Als men er steeds van uitgegaan is, dat variabelen alleen maar getallen kunnen voorstellen, dan komt bovenstaande wellicht wat vreemd over.
Een reeks schrifttekens noemt men een string en een variabele waaraan men zo'n string kan toekennen noemt men heel toepasselijk een string-variabele.
Om onderscheid te maken tussen de in de voorgaande hoofdstukken behandelde variabelen, die we nu maar getal-variabelen zullen noemen, en de nu te behandelen string-variabelen, wordt achter de naam van de laatste categorie steeds een dollar-teken geplaatst.
De algemene regel voor het geven van een naam aan een string-variabele luidt:

Een naam van een string-variabele mag slechts uit één letter bestaan en wel gevolgd door het dollarteken: $.

Voorbeelden van correct gespelde namen voor string-variabelen zijn: A$, C$, Z$ en P$.
Alhoewel de meeste BASIC-versies ruimere mogelijkheden met betrekking tot het geven van namen van string-variabelen bieden, leiden de regels bij de ZX81 toch niet tot namen die in tegenstrijd zijn met de algemeen geldende regels.
Om te tonen dat string-variabelen echte 'variabelen' zijn, waar men dus verschillende reeksen schrifttekens aan kan toekennen, tonen we het nu volgende programma:

```
10 LET A$="KLAAS"
20 LET B$=" VAAK"
30 LET C$=A$+B$
40 PRINT "DE NAAM IS ";C$
50 LET A$="WELTERUSTEN"
60 PRINT A$
```

Dit programma geeft als resultaat:

```
DE NAAM IS KLAAS VAAK
WELTERUSTEN
```

In regel 10 wordt aan de string-variabele A$ de waarde KLAAS toegekend. Merk op dat de string tussen de bekende aanhalingstekens wordt geplaatst. Dit plaatsen van aanhalingstekens is verplicht. De toekennings-instructie start weer met de specifieke term LET. In regel 20 krijgt B$ de string VAAK toegekend. Dat we met dergelijke strings echt kunnen manipuleren blijkt uit de volgende regel. Hier wordt aan de variabele C$ de string A$ + B$ toegekend.
Het teken + heeft hier de betekenis van: rijg de twee strings aan elkaar. (zgn. concateneren) De daarop volgende PRINT-instructie bevat twee af te drukken zaken. In beide gevallen gaat het om een string, maar de eerste string is direct op de vertrouwde wijze in de PRINT-instructie opgenomen, terwijl de tweede string door middel van de string-variabele wordt aangeduid. Hierna wordt de oude waarde van A$ overschreven met een nieuwe waarde, nl. de string WELTERUSTEN. Het plusteken is het enige bewerkingsteken dat we bij strings mogen gebruiken. Hieruit mag men echter niet concluderen dat het aantal mogelijke manipulaties met strings dus wel gering zal zijn. Er zijn een aantal krachtige functies die men op strings kan toepassen.
De lengte van de string die men aan een string-variabele mag toekennen is niet begrensd, afgezien dan van de altijd aanwezige beperkingen van het geheugen.
In het nu volgende tonen we een aantal typische strings:

STRING	COMMENTAAR
"AB"	string bestaande uit 2 letters
"AB "	string bestaande uit 2 letters en één spatie. Merk op dat de spatie ook als een schriftteken wordt gezien.
" "	string bestaande uit drie spaties
"1254"	string bestaande uit 4 cijfers. De cijfers stellen hier dus een string voor en niet het getal 1254. We zullen zien dat er wel mogelijkheden zijn om de omzetting naar het getal te krijgen.
""	is een string die geen enkel schriftteken bevat. We zouden deze wel zeer bijzondere string met het getal 0 bij de getallen kunnen vergelijken.
""""""	string die bestaat uit twee aanhalingstekens. Merk op dat de buitenste aanhalingstekens het begin en het einde van de string markeren en de werkelijke string bevat dus alleen de twee getoonde aanhalingstekens. De binnenste tekens dienen met de Q-toets te worden ingevoerd.

In het nu volgende zullen we de verschillende functies bespreken die men op strings kan toepassen. Bij iedere functie wordt een illustratief voorbeeldprogrammaatje getoond.

LEN
Met de LEN-functie wordt het aantal schrifttekens in een string bepaald.
Bijvoorbeeld:

```
LEN "ZX81"
```

levert het getal 4; de string ZX81 telt immers 4 tekens.
Een voorbeeldprogramma:

```
10 PRINT "VOER EEN NAAM IN"
20 INPUT A$
30 LET LENGTE=LEN A$
40 PRINT "DEZE NAAM IS ";
50 PRINT LENGTE;
60 PRINT " SCHRIFTTEKENS LANG"
```

In regel 20 zien we een INPUT-instructie: nu moet echter een string worden ingevoerd. Na het commando RUN wordt eerst de tekst VOER EEN NAAM IN afgedrukt en vervolgens zien we twee aanhalingstekens om ons er aan te herinneren dat nu een string moet worden ingevoerd.

U kunt nu een naam (zonder de gebruikelijke aanhalingstekens) intypen.

Bijvoorbeeld:

```
JONNEKE
```

Hierna toont de computer:

```
DEZE NAAM IS 7 SCHRIFTTEKENS LANG
```

VAL
Als een string op een getal betrekking heeft kan het ook als zodanig worden omgezet,en wel met behulp van de VAL-functie.
Voorbeeld:

```
10 LET A$="1234"
20 LET B$=" HOEDJE VAN PAPIER"
30 LET C=VAL A$
40 LET D=1235-C
50 PRINT D;B$
```

Het resultaat is:

```
1 HOEDJE VAN PAPIER
```

In regel 10 wordt aan de string-variabele A$ de string 1234 (en dus niet het getal 1234) toegekend. Regel 20 spreekt voor zich. In regel 30 wordt de VAL-functie toegepast: het resultaat is dat aan de getalvariabele C (achter de naam C staat immers geen $-teken) de waarde 1234 wordt toegekend. Deze waarde wordt in de daarop volgende regel van het getal 1235 afgetrokken. De PRINT-instructie toont ons ten slotte het resultaat.
De VAL-functie kan ook op een uitdrukking worden toegepast, bijvoorbeeld:

```
PRINT VAL "2+SQR(9)"
```

geeft direct de waarde 5.

Als de VAL-functie zelf in een uitdrukking wordt gebruikt dient deze steeds voorop te staan. Zo is de instructie:

```
10 LET A=3+VAL "7"
```

niet correct: VAL"7" staat immers niet vooraan. De uitdrukking had moeten zijn:

```
10 LET A=VAL "7"+3
```

We merken hierbij nog op dat het argument van een string-functie zelf weer een string-functie mag bevatten.
Probeer onderstaande instructie maar eens uit:

```
PRINT VAL(STR$ LEN "TWEE")
```

Bij instructies die bepalen waar iets op het beeldscherm wordt geplaatst, zoals PRINT AT ... en de nog te bespreken functies PLOT en UNPLOT mag de VAL-functie alleen direct gebruikt worden om de eerste coördinaat aan te geven. Zo is PRINT AT VAL"3",4 correct, maar PRINT AT 3, VAL"4" is niet toegestaan, hier wordt de VAL-functie immers gebruikt om de tweede coördinaat aan te geven. (N. B. de twee waarden na de term AT noemt men de coördinaten: zie ook hoofdstuk 10.) Het beste is maar om de VAL-functie in het geheel niet bij dergelijke afdruk-instructies te gebruiken; dit voorkomt zeker vergissingen.

STR$
De STR$-functie heeft in feite het omgekeerde resultaat als de VAL-functie: nu wordt een getal in een string omgezet.
Voorbeeld:

```
10 LET A=1234
20 LET P$=" HOEDJE VAN PAPIER"
30 LET K$=STR$ A
40 LET M$=K$+P$
50 PRINT M$
```

Dit programma geeft als resultaat:

```
1234 HOEDJE VAN PAPIER
```

We zien hoe in regel 30 aan de string-variabele K$ de string "1234" wordt toegekend, en wel door de string-functie STR$ op de variabele A toe te passen. Het programma spreekt overigens voor zichzelf.
Het argument van de STR$-functie mag zelf ook weer een uitdrukking zijn. De string die de STR$-functie dan oplevert is de string die als resultaat zou verschijnen als die uitdrukking in een PRINT-instructie zou zijn opgenomen.

De STR$-functie komt bijvoorbeeld van pas als we een rij getallen zodanig onder elkaar willen plaatsen dat de 'komma's' (decimal points) keurig netjes onder elkaar komen te staan. Ieder getal wordt dan vóór het afdrukken door middel van een STR$-functie aan een string-variabele toegekend. Hierna kunnen we met behulp van de nog te bespreken slicing-techniek, schriftteken voor schriftteken uit de string lichten, om zo na te gaan of het om een decimal point gaat, en hoeveel schrift-

tekens voor deze decimal point geplaatst zijn. Aan de hand van de zo verkregen informatie kunnen we met behulp van een geschikt gekozen TAB-functie het getal bij het afdrukken precies positioneren.

CHR$ en CODE
Om de volgende twee functies te begrijpen dienen we te beseffen dat ieder schriftteken volgens een vaste afspraak met een bepaald codenummer overeenkomt. Deze afspraak geldt trouwens niet alleen voor de schrifttekens, maar ook voor alle andere symbolen die we afzonderlijk kunnen gebruiken. In appendix 6 treft men een volledig overzicht aan. Sla deze appendix er maar eens op na. Zo zien we bijvoorbeeld dat de letter A met codenummer 38 overeenkomt, of andersom codenummer 38 komt met letter A overeen.

Benieuwd hoe alle symbolen er op het beeldscherm uitzien? Probeer het volgende programma maar eens uit:

```
10 FOR K=0 TO 255
20 PRINT CHR$ K;
30 NEXT K
```

We merken hierbij nog op dat de codes bij de ZX81 volstrekt uniek zijn. Ze komen bijvoorbeeld niet overeen met de veel gebruikte ASCII-codes.
In het nu volgende zullen we zien dat de omzetting van code naar symbool met de CHR$-functie wordt verkregen en de omzetting schriftteken (symbool) naar code met de functie CODE.

CHR$
Met de functie CHR$ wordt een opgegeven codenummer in een daarmee corresponderend schriftteken (of symbool) omgezet. Als voorbeeld kunnen we naar het zojuist getoonde programma kijken. Aan de CHR$-functie worden hier achtereenvolgens de getallen 0 t/m 255 toegekend en door middel van de PRINT-instructie kan men dan zien met welk schriftteken (symbool) ieder getal correspondeert.
De CHR$-functie komt bijvoorbeeld van pas als we een speciaal schriftteken willen afdrukken wat we niet direct kunnen intoetsen. Door middel van de CHR$-functie kunnen we dan direct aangeven om wat voor teken het gaat. Ook kunnen we de CHR$-functie toepassen als afhankelijk van een bepaald resultaat een bepaald teken moet worden afgedrukt.

CODE
Na het voorafgaande zal de CODE-functie geen problemen meer opleveren. Het argument van de CODE-functie dient steeds een string te zijn. CODE zet dan het eerste teken van de string in het bijbehorende

codenummer om.
Bijvoorbeeld:

```
10 INPUT A$
20 LET K=CODE A$
30 PRINT K
```

Voeren we hier bijvoorbeeld de string "AAP" in dan zal het getal 38 worden getoond, omdat de eerste letter van de string "AAP", met andere woorden de letter "A" met het codenummer 38 overeenkomt. Merk op dat we aan de hand van de codenummers woorden ook in alfabetische volgorde kunnen plaatsen.

SLICING: de term TO
Bij zeer veel toepassingen komen we de vraag tegen om uit een bepaalde string een aantal tekens te lichten.
Zo kan men uit de string BASIC de volgende reeksen 'snijden':

B	A	S	I	C
BA	AS	SI	IC	
BAS	ASI	SIC		
BASI	ASIC			

We noemen deze strings: de sub-strings van de oorspronkelijk string "BASIC". In feite is er niets op tegen om ook de lege string "" en de volledige string "BASIC" als sub-string op te vatten.
Bij de ZX 81 verkrijgt men substrings met behulp van de 'slicing-techniek' ('uitsnij-techniek'). Hierin wijkt de BASIC van de ZX 81 duidelijk van andere BASIC-versies af. Bij de slicing-techniek wordt steeds na de string door middel van een uitdrukking tussen twee haakjes, aangegeven om wat voor substring het gaat.
In het algemeen treft men de volgende uitdrukking aan:

string (begin TO eind)

Voorbeeld:

"COMPUTER"(2 TO 4) levert de string "OMP"

Onderstaande tabel toont een aantal andere voorbeelden.

Uitdrukking	Resultaat	Opmerkingen
"COMPUTER" (1 TO 4)	"COMP"	de eerste 4 letters
"COMPUTER" (TO 4)	"COMP"	laat men de begin-aandui-

Uitdrukking	Resultaat	Opmerkingen
		duiding weg dan wordt automatisch 1 verondersteld.
"COMPUTER"(6 TO 8)	"TER"	de laatste 3 letters
"COMPUTER"(6 TO)	"TER"	laat men de eind-aanduiding weg dan wordt automatisch de laatste letter verondersteld
"COMPUTER"(1 TO 8)	"COMPUTER"	substring bestaat uit alle letters, kortom uit de gehele string
"COMPUTER"(TO)	"COMPUTER	laat men begin- en eind-aanduiding weg dan verkrijgt men de gehele string als resultaat
"COMPUTER"(10 TO 11)	""	de string telt maar 8 schrifttekens: we vinden dus de lege string als resultaat
"COMPUTER"(4)	"P"	op deze wijze wordt alleen de vierde letter uit de string gesneden

Ten slotte nog een programma-voorbeeld:

```
10 LET A$="COMPUTER"
20 LET AANTAL=LEN A$
30 FOR K=1 TO AANTAL
40 LET B$=A$(1 TO K)
50 PRINT B$
60 NEXT K
```

Dit programma geeft als resultaat:

```
C
CO
COM
COMP
```

```
COMPU
COMPUT
COMPUTE
COMPUTER
```

Merk op hoe in regel 20 door middel van de LEN-instructie aan AANTAL het aantal schrifttekens van de string "COMPUTER" wordt toegekend. Deze grootheid wordt bij de slicing in regel 40 toegepast. Aangezien regel 40 deel uitmaakt van een FOR ... NEXT-instructie zien we de getoonde substring verschijnen.

We kunnen ook aan de hand van een LET-instructie aan een substring een reeks schrifttekens toekennen.
Voorbeeld:

```
10 PRINT "RAADSPELLETJE"
20 INPUT A$
30 LET LENGTE=LEN A$
40 LET B$=A$
50 LET B$(2 TO LENGTE-1)="**********"
60 CLS
70 PRINT B$
80 PRINT "WELK WOORD IS DIT?"
90 INPUT W$
100 IF W$=A$ THEN GOTO 140
110 PRINT "FOUT.... HET WOORD WAS"
120 PRINT A$
130 STOP
140 PRINT "GERADEN "
```

Dit spelletje dient met twee personen gespeeld te worden. De eerste persoon voert een naam in en de tweede persoon ziet alleen maar de eerste en de laatste letter. Op grond van deze informatie dient het woord dan geraden te worden. We zien hoe in regel 20 aan de string-variabele A$ het ingevoerde woord wordt toegekend. In de daarop volgende regel wordt met de LEN-instructie het aantal schrifttekens bepaald. Vervolgens wordt de string ook aan B$ toegekend. De inhoud van deze string-variabele gaan we vervolgens sterk verminken. Dat gebeurt dan in regel 50; hier worden de tweede tot en met de een na

laatste letter door sterretjes vervangen. De sterretjes-string, rechts van het =-teken,bevat 10 tekens, dus men kan zich met recht afvragen: wat gebeurt er als dit ten opzichte van de aangegeven lengte (2 TO LENGTE-1) te veel of te weinig tekens zijn?
Het antwoord luidt dat bij te veel tekens automatisch tekens worden afgekapt en wel van rechts, en bij te weinig tekens worden er automatisch (aan de rechterkant) spaties toegevoegd. Men spreekt in dit geval van een Procrusteaanse toekenning. Procrustes was een herbergier die er voor zorgde dat de gasten precies in het bed pasten door ze uit te rekken of door hun voeten af te snijden ...
In regel 60 wordt het beeldscherm gewist door de CLS-instructie toe te passen.
Regel 70 tot en met 140 beschrijven verder een gedeelte, dat geen verdere uitleg meer behoeft.

Arrays met strings
In hoofdstuk 15 werd het begrip array uiteengezet. Er werden behalve de 'gewone' arrays ook de meer-dimensionale arrays besproken.
We brengen weer in herinnering dat bij de instructie:

```
DIM A(2,3)
```

ruimte wordt gereserveerd voor de volgende variabelen:

```
A(1,1)   A(1,2)   A(1,3)
A(2,1)   A(2,2)   A(2,3)
```

Deze variabelen kunnen als iedere andere variabele gebruikt worden, bijvoorbeeld:

```
LET B=A(2,1)+4
```

De ZX81 kent ook de mogelijkheid om arrays met strings te gebruiken. Dit zijn dan altijd meer-dimensionale arrays met dien verstande dat iedere afzonderlijke variabele maar één schriftteken kan bevatten. Achter de naam van de array dient weer het vertrouwde $-teken te staan.
Voorbeeld van een DIM-instructie bij een string-array:

```
DIM A$(2,3)
```

In feite vormt een rij string-array-variabelen (variabelen die voor de komma een gelijk nummer hebben), zoals:

```
A$(1,1)   A$(1,2)   A$(1,3)
```

één string en omdat deze rij uit drie variabelen bestaat kan deze rij een woord van precies drie letters bevatten. We kunnen een bepaalde

rij eenvoudig aangeven met het rij-nummer, bijvoorbeeld:

```
LET A$(1)="BAS"
```

heeft tot gevolg dat aan de rij A$(1,1), A$(1,2) en A$(1,3) respectievelijk de letters B, A en S zijn toegekend.
We illustreren een en ander nogmaals aan de hand van een voorbeeld:

```
10 DIM X$(3,4)
20 FOR K=1 TO 3
30 INPUT X$(K)
40 NEXT K
50 FOR K=3 TO 1 STEP -1
60 PRINT X$(K)
70 NEXT K
```

Voert men achtereenvolgens in:

```
HOND
POES
MUIS
```

dan antwoordt de computer met:

```
MUIS
POES
HOND
```

Ten slotte merken we op dat ook de reeds besproken slicing-techniek mag worden toegepast.
Bijvoorbeeld:

```
LET B$=X$(2)(3 TO 4)
```

Als aan X$(2) de string "POES" was toegekend zal B$ gelijk worden aan "ES".

18. Over plaatjes

Een van de zaken die een computer van een calculator onderscheidt is het feit dat we met een computer allerlei plaatjes op een beeldscherm kunnen 'tekenen'. Deze mogelijkheid komt uiteraard bij zeer veel praktische toepassingen van pas, denk maar eens aan de grafische weergave van resultaten en het afbeelden van figuren bij onderwijskundige toepassingen. Ook bij veel spelletjes zijn plaatjes onontbeerlijk. Wie tegenwoordig langs een automatenhal loopt ziet ongetwijfeld de vele machines waar bijvoorbeeld raketten gillend over het beeldscherm worden afgeschoten. Allemaal zaken die zonder de computer van nu niet mogelijk zouden zijn.
De ZX81 biedt tevens de mogelijkheid om plaatjes weer te geven. We kunnen zelfs bewegende plaatjes verkrijgen. In folders spreekt men vaak van 'graphics' als men het heeft over het afbeelden van figuren, en het feit dat zelfs bewegende patronen kunnen worden weergegeven, wordt aangeduid met 'animated graphics' (vertaald: plaatjes die boeien).
Om bij spelletjes de mogelijkheid te krijgen onmiddellijk op bepaalde situaties te kunnen reageren, is bij de ZX81 nog een afzonderlijke instructie toegevoegd, nl. de INKEY$-instructie.
De volgende instructies zijn bij het werken met plaatjes van belang:

PRINT AT — Voor de weergave van schrifttekens en grafische symbolen op iedere willekeurige plaats van het beeldscherm.

PLOT — Voor het tekenen van figuren: in feite voor de weergave

van een klein zwart vierkantje op een aangegeven plaats. Door veel van deze vierkantjes naast elkaar te plaatsen, kan men lijnen of andere figuren verkrijgen.

UNPLOT — Voor het wissen van reeds weergegeven zwarte vierkantjes. In combinatie met PLOT kan men zelfs bewegende beelden verkrijgen. Zo kan men een punt plotten en daarnaast weer een punt terwijl het vorige punt weer gewist wordt enz. Dit levert uiteindelijk een bewegend puntje op.

PAUSE — Om een zelf te bepalen wachttijd te introduceren.

INKEY$ — Instructie om een momentane toetsindruk in te lezen. Vooral interessant om onmiddellijk op bepaalde situaties te kunnen reageren.

SLOW — Voor het weergeven van plaatjes tijdens het rekenen. In deze toestand rekent de computer relatief traag. (Werkt niet bij de ZX80 met uitgebreide BASIC-versie.)

FAST — Voor het snelle rekenwerk. Tijdens het rekenen dooft het beeldscherm.

In het nu volgende zullen we dieper op deze instructies ingaan. Bij de PLOT-instructie zal iets meer worden verteld over het afbeelden van lijnen. Aan het eind van dit hoofdstuk zal getoond worden hoe men bewegende objecten kan verkrijgen.

PRINT AT

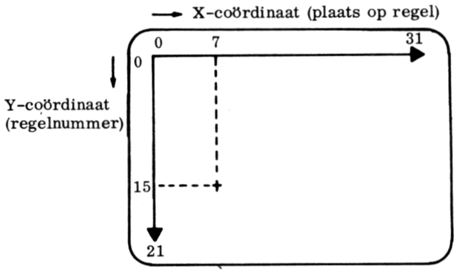
De instructie PRINT AT werd reeds in hoofdstuk 10 besproken. De gedaante van deze instructie is steeds als volgt:

PRINT AT regelnummer, plaats op regel;"tekst"

Bijvoorbeeld:

```
PRINT AT 15,7;"+"
```

Deze instructie heeft tot gevolg dat het symbool + op het beeldscherm wordt afgebeeld, en wel op de coördinaten 15 en 7.
Onderstaande figuur toont voor de duidelijkheid de indeling van het beeldscherm, zoals die bij de PRINT-instructies gebruikt wordt.



We zien dat met de PRINT AT-instructie alle plaatsen met een X-coördinaat van 0 t/m 31 en een Y-coördinaat van 0 t/m 21 bereikt kunnen worden. Reeds beschreven plaatsen kunnen weer worden overschreven.
Onder het aangegeven vlak zijn nog twee regels vrij om tekst op het beeldscherm af te beelden.

Met name de nog niet besproken grafische symbolen zijn interessant in verband met het afbeelden van figuren. Deze symbolen kunnen alleen maar met een PRINT AT-instructie op een aangegeven plaats worden afgebeeld. Bij de nog te bespreken PLOT-instructie kan men alleen maar kleine zwarte vierkantjes afbeelden.
Hier volgt een overzicht van de grafische symbolen:

symbool	code	symbool	code
	0		128
	1		129
	2		130
	3		131
	4		132
	5		133
	6		134
	7		135
	8		136
	9		137
	10		138

We zien naast ieder grafisch symbool de bijbehorende code. Deze grafische symbolen treft men ook op het toetsenbord aan. Zo vindt men bij de toets A het grafische symbool 'vierkantje met schaakbordpatroon'.
Een grafisch symbool toetsen we op de volgende wijze in:

- Druk de toets met de aanduiding GRAPHICS in, dat wil zeggen houdt de toets SHIFT ingedrukt en druk vervolgens op de toets met het cijfer 9.
- Hierna verschijnt de cursor met de letter G (van Graphics).
- Druk hierna SHIFT in en druk vervolgens op een van de toetsen, waarbij een grafisch symbool staat (bijvoorbeeld op toets A als we het schaakbordpatroon willen zien).
- Nu kunnen we meer grafische symbolen intoetsen en wel tot het moment dat GRAPHICS of NEWLINE weer wordt ingedrukt.
- De enige uitzondering op bovenstaande vormt het spatie-teken (code 0) dat direct kan worden ingevoerd door op SPACE te drukken.

Uiteraard kunnen we ieder symbool ook indirect door middel van zijn codenummer aanduiden. Dit wordt in het volgende programma getoond:

```
10 FOR K=128 TO 138
20 LET A$=CHR$ K
30 PRINT A$;
40 NEXT K
```

Op deze wijze verkrijgen we een balk die bestaat uit de symbolen met de codes 128 t/m 138.
We merken ten slotte nog op dat als op een toets gedrukt wordt terwijl de cursor een G (van Graphics) aangaf het desbetreffende symbool in wit tegen een zwarte achtergrond wordt afgebeeld.

PLOT

De instructie PLOT heeft steeds de volgende gedaante:

PLOT X-coördinaat, Y-coördinaat

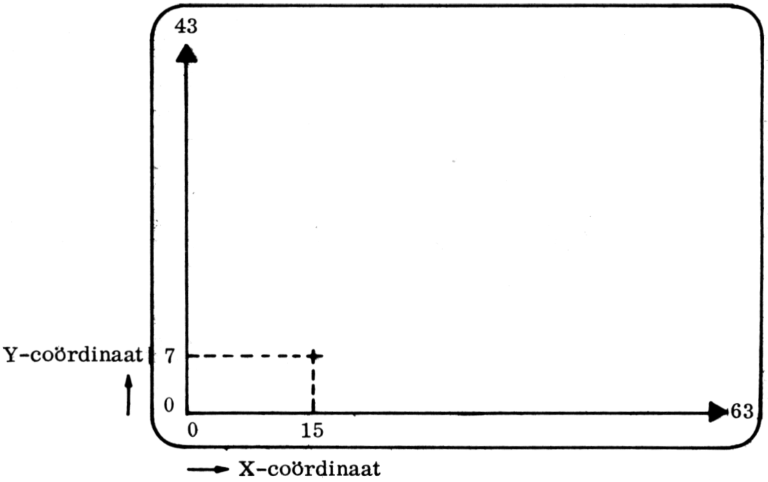
Bijvoorbeeld:

```
PLOT 15,7
```

Als gevolg hiervan zal een zwart vierkantje ontstaan op het beeldscherm en wel op de coördinaten 15 en 7. Pas echter op!

DE COORDINATEN BIJ DE PLOT- EN UNPLOT-INSTRUCTIE ZIJN ANDERS DAN BIJ DE PRINT AT-INSTRUCTIE. BIJ PRINT AT IS HET BEELDSCHERM VERDEELD IN 32 X 22 BLOKKEN. BIJ PLOT (EN UNPLOT) IS HET BEELD VERDEELD IN 64 X 44 BLOKKEN. BOVENDIEN LIGT DE 0-COORDINAAT VAN DE Y-AS ANDERS

De X-coördinaat bij een PLOT-instructie loopt van 0 t/m 63 en de Y-coördinaat loopt van 0 t/m 43 en wel volgens de volgende afspraak:



Merk op dat het 0-punt van de Y-coördinaat nu onderaan ligt, terwijl dit bij de PRINT AT-instructie juist bovenaan ligt.
Men kan aan de hand van het volgende programma de nodige ervaring voor beide instructies opdoen:

```
10 PRINT "X-COORD. V. PLOT"
20 INPUT XPLOT
30 PRINT "Y-COORD. V. PLOT"
40 INPUT YPLOT
50 PRINT "X-COORD. V. PRINT AT"
60 INPUT XPRINT
70 PRINT "Y-COORD. V. PRINT AT"
```

```
80 INPUT YPRINT
90 PRINT AT XPRINT,YPRINT;"0"
100 PLOT XPLOT,YPLOT
```

We kunnen met behulp van de PLOT-instructie eenvoudig rechte lijnen afbeelden.
Het volgende programma toont hoe men een willekeurige horizontale lijn (dat wil zeggen Y-coördinaat is constant) kan afbeelden:

```
10 PRINT "VOER Y-COORDINAAT IN"
20 INPUT Y
30 FOR K=1 TO 63
40 PLOT K,Y
50 NEXT K
```

Dit programma draait om regel 40. De instructie PLOT K, Y heeft tot gevolg dat aaneensluitende vakjes met een vaste Y-coördinaat worden afgebeeld: kortom, we zien een horizontale lijn en wel ter hoogte van de ingevoerde Y-coördinaat.
Evenzo voor een verticale lijn:

```
10 PRINT "VOER X-COORDINAAT IN"
20 INPUT X
30 FOR K=1 TO 43
40 PLOT X,K
50 NEXT K
```

Dit programma is vrijwel gelijk aan het vorige programma.

Een schuine lijn kunnen we verkrijgen door zowel de X- als de Y-coördinaat met vaste stappen in een FOR ... NEXT-instructie te wijzigen. Bijvoorbeeld:

```
10 PRINT "VOER HELLING IN"
20 INPUT M
30 FOR K=1 TO 63
40 LET XCOR=K
50 LET YCOR=M*XCOR
60 IF YCOR>43 THEN GOTO 80
70 PLOT XCOR,YCOR
80 NEXT K
```

Probeer het programma maar eens uit voor de waarde M = 1.
Merk op dat als gevolg van regel 40 de X-coördinaat steeds met stappen van 1 wordt gewijzigd en de Y-coördinaat met stappen van M *XCOR, met andere woorden met stappen die M keer zo groot zijn dan de stappen van de X-coördinaat. De Y-coördinaat mag niet groter dan 43 worden (43 is nl. de rand van het beeldscherm) en om dit te waarborgen is regel 60 opgenomen.
In dit geval loopt de lijn nog door het punt 0,0 (zie vorige figuur). We noemen dit punt de 'oorsprong'. Men kan de lijn ook nog optillen, zodat de lijn niet meer door de oorsprong gaat, en wel door regel 50 als volgt te wijzigen.

```
50 LET YCOR=M*XCOR+B
```

waarbij B van te voren een of andere constante waarde heeft gekregen, bijvoorbeeld door middel van een INPUT-instructie. Probeer dit uit! Uiteraard kunnen we allerlei lijnen en curven uitbeelden. Bijvoorbeeld (voorbeeld uit Engelse handleiding):

```
10 FOR N=0 TO 63
20 PLOT N,22+20*SIN(N/32*3.14)
30 NEXT N
```

heeft tot gevolg dat we een sinus-curve op het beeldscherm zien. Het is aardig om dit programma met het programma van bladzijde 73 te vergelijken.

PAUSE

De PAUSE-instructie wordt gebruikt om het rekenproces tijdelijk te onderbreken. Na PAUSE noteert men steeds een getal om de wachttijd aan te geven. Om de gedachten te bepalen: het getal 50 komt met een wachttijd van ca. 1 seconde overeen.
Als voorbeeld tonen we een programma dat van uw computer een digitaal uurwerk maakt:

```
10 FOR U=0 TO 23
20 FOR M=0 TO 59
30 FOR S=0 TO 59
40 PRINT U;" UUR ";M;" MINUTEN"
50 PRINT "EN ";S;" SECONDEN"
60 PAUSE 50
65 POKE 16437,255                (alleen voor ZX80)
```

```
70 CLS
80 NEXT S
90 NEXT M
100 NEXT U
```

Het programma bestaat uit drie FOR ... NEXT-instructies. De buitenste lus telt de uren, dan volgt een binnenlus waarmee de minuten worden geteld en ten slotte volgt een lus om de seconden te tellen. De binnenlus (regel 30 tot en met 80) bevat de nodige PRINT-instructies en de bewuste PAUSE-instructie.
We merken hierbij nog op, dat bij de ZX80 met uitgebreide BASIC na de PAUSE-instructie altijd de volgende POKE-instructie moet worden vermeld:

```
POKE 16437,255
```

Bijvoorbeeld in bovenstaand programma (tussen regel 60 en 70):

```
65 POKE 16437,255
```

INKEY$
Als de instructie INKEY$ wordt aangeroepen wordt momentaan gekeken welke toets wordt ingedrukt. In feite kunnen we INKEY$ het beste als een string-variabele beschouwen, die precies één schriftteken kan bevatten. Zodra in een programma een regel waarin INKEY$ voorkomt wordt afgewerkt, wordt het toetsenbord bekeken en de dan ingedrukte toets wordt vervolgens aan de 'variabele' INKEY$ toegekend. Deze waarde kunnen we dan verder als een gewone string, bestaande uit één schriftteken, behandelen.
Als geen enkele toets wordt ingedrukt op het moment dat INKEY$ in het programma wordt aangetroffen, zal INKEY$ gelijk worden aan de lege string "".
Het volgende programma is illustratief:

```
10 PRINT INKEY$;
20 GOTO 10
```

(N. B. bij ZX80 met uitgebreide BASIC-versie toevoegen: 15 PAUSE 50 en 17 POKE 16437,255.)
Bij het uitvoeren van dit programma zal voortdurend de toets worden afgebeeld die u momentaan indrukt. Regel 10 geeft aan dat de toets die u indrukt en die dus aan de 'variabele' INKEY$ wordt toegekend, dient te worden afgedrukt.
Drukt u geen toets in dan zal de lege string "" worden afgebeeld met

als resultaat dat er op het beeldscherm niets verandert.

Het volgende programma illustreert INKEY$ in een IF-instructie:

```
10 PRINT "*";
20 IF INKEY$="S" THEN STOP
30 GOTO 10
```

Als gevolg van dit programma zal het beeldscherm met sterretjes gevuld worden en wel tot het moment dat u op de toets S drukt. Als gevolg van de instructie op regel 20 zal de computer dan met het uitvoeren van het programma stoppen.

SLOW en FAST

Met de toetsen SLOW en FAST kunnen we de computer in de trage (SLOW) en snelle (FAST) modus laten rekenen. Het verschil in rekensnelheid bedraagt ca. een factor 4.
In de modus FAST dooft het beeld tijdens het rekenen en bij het intoetsen van een programma flikkert het beeld bij iedere toetsindruk. Het beeld zal alleen voortdurend blijven oplichten als het einde van het rekenwerk is bereikt of als een PAUSE-instructie wordt aangetroffen. Tegenover het verlies in snelheid staat het gemak van een voortdurend oplichtend beeld als de computer in de SLOW-modus rekent. Dit is op de ZX 80 met uitgebreide BASIC-versie niet mogelijk.
Zet men de ZX 81 aan dan komt deze automatisch in de SLOW-modus. Het zal duidelijk zijn dat de FAST-modus te prefereren is als er veel gerekend moet worden. Bij spelletjes zal men daarentegen meestal de SLOW-modus gebruiken.

Bewegende figuren

Niets is leuker dan bewegende figuren ... levende figuren. Het principe van het bewegen komt overeen met het principe van de bewegende film. Door een groot aantal plaatjes snel na elkaar te tonen ontstaat de indruk van een bewegend beeld, tenminste als de figuur op deze plaatjes steeds iets verschoven staat. Dit houdt in dat we bewegende plaatjes kunnen krijgen door:

a. een figuur af te beelden
b. dit figuur een kort poosje te tonen
c. dit figuur weer te wissen
d. het figuur weer te tonen maar dan op een iets verschoven positie enz.

Als wis-functies kunnen de CLS- en de UNPLOT-instructie dienen. Het volgende programma toont een eenvoudig voorbeeld:

```
10 INPUT SNELHEID
```

```
20 FOR K=0 TO 31
30 PRINT AT 10,K;"■"
40 PAUSE SNELHEID
45 POKE 16437,255          (alleen voor ZX80)
50 CLS
60 NEXT K
```

(N. B. ■ intoetsen door eerst op GRAPHICS te drukken, en vervolgens op SPACE, hierna weer op GRAPHICS drukken om " in te toetsen)
In regel 10 staat een INPUT-instructie, waaruit volgt dat u een waarde moet invoeren die aan SNELHEID wordt toegekend. Dan volgt een FOR ... NEXT-instructie; steeds wordt het zwarte vierkantje afgebeeld (regel 30), een poosje getoond (regel 40 en 45) en weer gewist (regel 50). Merk op dat de K-coördinaat steeds groter wordt; we krijgen de illusie dat het zwarte vlakje over het beeldscherm beweegt.

In het nu volgende programma wordt een hele reeks zwarte vierkantjes afgebeeld, en wel zodanig dat het beeld van een vliegtuig ontstaat. De X-coördinaat van al deze PRINT-instructies wordt tevens in een FOR ... NEXT-instructie steeds opgehoogd. Zo zal de indruk ontstaan dat het vliegtuig op het beeldscherm vliegt.
Het is de bedoeling het vliegtuig neer te halen door op één van de toetsen 1 t/m 8 te drukken. De waarde 1 komt met de positie 'geheel links' overeen en 8 met de positie 'geheel rechts'. In het geheel zullen 10 vliegtuigen overvliegen. Het programma is zo ontworpen dat u per vliegtuig maar één keer kunt schieten.
Als u raak schiet verschijnt:

```
RAAK
```

op het beeldscherm.
Om het gehele programma in 1K geheugen te persen is zuinig met tekst omgesprongen. Na het commando RUN verschijnt de vraag:

```
V=
```

op het beeldscherm. Hierna kunt u een getal intoetsen waarmee de vliegsnelheid geregeld wordt.
Na afloop verschijnt het aantal treffers.
Ten slotte nog een hint voor het intoetsen: de zwarte blokken krijgen we door de computer in de GRAPHICS-modus te plaatsen en door hierna op SPACE te drukken. Voor een gewone spatie daarna (hier aangeduid met "b") moet men eerst weer op GRAPHICS drukken (de computer verandert dan weer van modus) en spaties worden dan weer

'witte' vierkantjes.

PROGRAMMA VLIEGTUIGSCHIETEN

```
 2 PRINT "V="
 4 INPUT SN
 5 LET A=0
 6 RAND
 8 FOR K=1 TO 10
10 CLS
12 LET VL=0
14 LET W$=""
16 LET R=INT((RND*15)+1.)
18 FOR X=0 TO 24
20 GOSUB 60
22 PAUSE SN
24 POKE 16437,255                    (alleen voor ZX80)
26 CLS
28 IF INKEY$="" THEN GOTO 48
30 IF VL=1 THEN GOTO 48
32 LET W$=INKEY$
34 LET VL=1
38 LET U=3*VAL W$
40 IF ABS(X-U)>2 THEN GOTO 48
42 PRINT AT 0,0;"GERAAKT"
43 LET A=A+1
44 PAUSE 100
45 POKE 16437,255                    (alleen voor ZX80)
46 GOTO 50
48 NEXT X
50 NEXT K
52 PRINT A
```

```
54 STOP
60 PRINT AT R,X;"bb■"
62 PRINT AT R+1,X;"■bb■■"
64 PRINT AT R+2,X;"■■■■■■"
66 PRINT AT R+3,X;"bbb■■"
68 PRINT AT R+4,X;"bb■"
70 RETURN
```

(N. B. De letter b wordt hier gebruikt om een spatie aan te geven.)

Wellicht dat de resultaten toch nog iets tegenvallen. De ZX 81-bezitters kunnen nagaan wat er gebeurt als regel 22 (met de PAUSE-instructie) wordt weggelaten.
Het programma om een horizontale lijn te trekken (blz. 99) toont in feite wat de optimale mogelijkheden zijn om m.b.v. BASIC-instructies bewegende objecten te krijgen. Probeer dit programma ook maar eens uit waarbij in iedere lus het voorafgaande punt gewist wordt.
De combinatie PLOT en UNPLOT kan tot zeer fraaie effecten leiden. Echt optimale prestaties t.a.v. het bereiken van bewegende beelden kunnen we met machine-instructies behalen.

19. Cassetterecorder en printer

Apparaten die men aan een computer kan koppelen worden gewoonlijk met de term 'perifere apparaten' aangeduid.
In dit hoofdstuk bespreken we:
de cassetterecorder: Voor het opslaan van een programma op tape. Zo hoeft men een volgende keer het programma niet meer opnieuw in te toetsen, maar men kan volstaan met het 'laden' van het programma van de cassetterecorder. De cassetterecorder dient ook gebruikt te worden voor het afspelen van reeds voorbespeelde cassettes.
de printer: Voor het afdrukken van tekst en meer in het algemeen, voor het afdrukken van het beeldscherm op papier.

Cassetterecorder
Bij de ZX81 zijn reeds snoertjes voor de aansluiting met een cassetterecorder bijgeleverd. Gebruik een gewone mono (bijv. draagbare) cassetterecorder, liefst met teller en gebruik 'low noise' tapes.

Het laden van een programma vanuit het geheugen op tape verloopt als volgt:

1. Sluit de kabel aan: verbind EAR (plug op computer) met EAR (plug op cassetterecorder) en MIC (plug op de computer) met MIC (plug op de cassetterecorder). Als de draden per ongeluk verwisseld worden, zal geen schade ontstaan.
2. Plaats de tape op een nog niet bespeeld gedeelte. Van te voren kan eventueel nog, vóór dit gedeelte, de naam van het programma door de microfoon worden ingesproken. Dit vereenvoudigt het zoeken achteraf.
3. Stel ter illustratie dat we het volgende, zeer korte programma in het geheugen hebben staan:

   `10 REM DIT IS DE ENIGE REGEL` (gevolgd door NEWLINE)

   dan moeten we voor het wegschrijven van dit programma het commando SAVE geven , en wel gevolgd door een vrij te kiezen naam. Druk in geen geval bij dit SAVE-commando NEWLINE in. Bijv. :

   `SAVE "REGEL"` (en druk nog niet op NEWLINE)

4. Start de recorder zodat deze opneemt (gewoonlijk RECORD indrukken en in deze stand op PLAY drukken).
5. Druk nu pas op NEWLINE.

6. Eerst toont de TV een grijsachtig beeld, dan ziet men allerlei strepen en soms horen we de meest afgrijselijke geluiden ...
   Wanhoop echter niet. Na een poosje dient op het beeldscherm de boodschap 0/0 te verschijnen.
7. Pas bij het weer laden van het programma kunnen we nagaan of de opname wel goed is geweest.

Het laden van een programma van cassette naar geheugen van de computer geschiedt als volgt:

1. Verzeker u ervan dat het snoertje dat EAR verbindt goed is aangesloten. (N. B. soms gebruikt men de luidsprekeruitgang van de cassetterecorder omdat deze in een aantal gevallen een wat krachtiger signaal produceert.)
2. Plaats het bandje in de goede positie en plaats de volumeknop in een stand van ca. 3/4 van het maximum. Indien een toonhoogte-regeling aanwezig is, plaats deze dan in de stand 'hoge tonen'.
3. Type in:

   LOAD naam van het programma

   bijvoorbeeld:

   `LOAD REGEL`   (en nog niet op NEWLINE)

   Men kan de naam eventueel weglaten. De computer leest dan het eerste programma in, dat bij het afspelen wordt aangetroffen.
4. Laat de cassetterecorder afspelen (druk op PLAY) en druk hierna pas op NEWLINE.
   Wederom zien we vreemde beelden en horen we soms vreselijke geluiden. Op een aantal recorders kan men dit geluid afzetten, hetgeen wellicht is aan te bevelen om ruzie in huiselijke kring te voorkomen.
5. Na een poosje dient het bericht 0/0 te verschijnen. Als het niet lukt onderbreekt u het inlezen met het commando BREAK. In dat geval probeert u de procedure opnieuw, waarbij u uittest hoe de knoppen moeten staan. Eenmaal gelukt ... noteer de stand dan goed.

Enkele tips:
In zeer veel gevallen gaat het opnemen en uitlezen op cassetterecorder de eerste malen mis.
Het is meestal een kwestie van uitproberen.
Hier volgen een aantal tips:

- soms moet men de luidsprekeruitgang van de recorder gebruiken
- maak eventueel de koppen van de recorder schoon (gebruik de speciaal hiervoor ontwikkelde reinigingsbandjes)
- bij het afluisteren van de band dienen de stille gedeelten in de band net niet ruis-achtig te klinken
- soms dient bij het opnemen op band de 'EAR-verbinding' losgemaakt te worden.

De printer

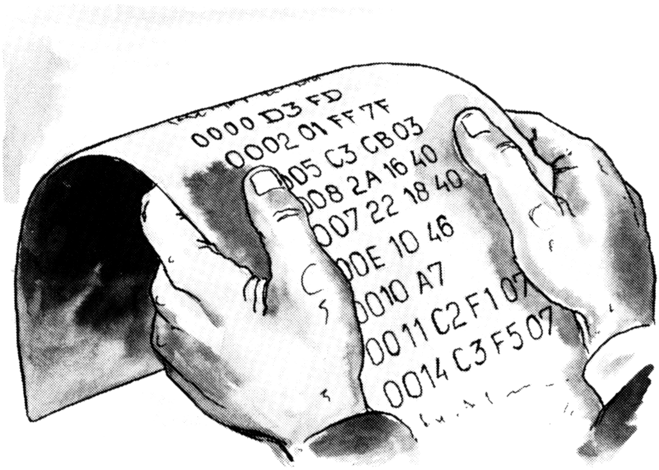
De firma Sinclair levert een afzonderlijke printer (afdrukapparaat) voor de ZX 81. Hiervoor zijn de volgende instructies/commando's beschikbaar:

LLIST: als LIST maar nu wordt alle tekst op de printer afgedrukt

LPRINT: als PRINT maar nu wordt de printer gebruikt.

Ten slotte kent men de instructie COPY als gevolg waarvan een copie van het beeldscherm zal worden afgedrukt.

20. Machine-instructies



De opzet van dit boek is in eerste instantie, de lezer duidelijk te maken hoe men met behulp van de programmeertaal BASIC allerlei zaken op de ZX81 kan uitvoeren.
In de meeste gevallen zal de ZX81 inderdaad alleen maar als 'BASIC-computer' gebruikt worden.
Het programmeren met behulp van machine-instructies strookt nauwelijks met bovengenoemde doelstelling en we zullen er dan ook niet diep op ingaan.
Bedenk vooral dat het programmeren met behulp van machine-instructies voor een beginneling vaak een zeer moeilijke zaak is. Het is ten ene male ondoenlijk om het fijne van machine-instructie-programmering in een paar bladzijden uit te leggen. De geïnteresseerden verwijzen we naar 'Programming the Z 80' van Rodnay Zaks (uitgave Sybex).
We volstaan hier met de volgende opzet:

a. allereerst beschrijven we wat nu wel het programmeren met behulp van machine-instructies inhoudt
b. vervolgens gaan we in op de vraag 'wanneer is het nodig om machine-instructies te gebruiken?'
c. en ten slotte behandelen we de instructies PEEK, POKE en USR.

Wat zijn machine-instructies

Machine-instructies zijn de meest elementaire instructies die de computer kent; het zijn in feite de enige instructies die de computer be-

grijpt. Ieder programma (dus ook al onze BASIC-programma's) worden uiteindelijk in deze machine-instructies omgezet. Deze omzetting geschiedt grappig genoeg door ... een programma in machine-instructies.
In feite zijn machine-instructies rijtjes enen en nullen. Om een meer compacte notatie te verkrijgen, hanteert men vaak een talstelsel met de symbolen 0 t/m 9 en A t/m F (zgn. hexadecimale stelsel). Op deze wijze kan men ieder rijtje enen en nullen eenvoudig weergeven. Om de acties van de machine-instructies beter te laten uitkomen is een taal ontwikkeld die zeer dicht bij het niveau van de machine-instructies staat. Dit is de assembleertaal. De ZX 81 beschikt over een Z 80-processor (rekenorgaan) en dit houdt in dat alleen voor de Z 80 ontworpen assembleertalen voor ons interessant zijn.

Wanneer machine-instructies?

Ontwerpers van computer-systemen hebben meestal met machine-instructies te maken. In de ZX 81 is een programma in ROM-geheugen (zie hoofdstuk 1) 'ingebakken' dat onze BASIC-instructies in machine-instructies omzet. Dit programma is in de assembleertaal geschreven.
In het algemeen gebruikt men machine-instructies als:
- de taal BASIC niet meer toereikend is, bijv. bij het koppelen van zeer speciale apparaten
- een programma optimaal snel moet zijn; sneller dan een oplossing met machine-instructies is niet mogelijk
- men in bestaande programma's, bijv. in het aanwezige BASIC-programma, wil ingrijpen.

Uit deze voorbeelden kan men afleiden dat machine-instructies in feite geheel en al buiten het kader van dit boek vallen.
BASIC biedt niettemin een aantal instructies waarmee de koppeling naar machine-instructies tot stand gebracht kan worden. Deze worden in het nu volgende kort behandeld.

De instructies POKE, PEEK en USR

Om tot het niveau van machine-instructies af te dalen is het nodig om direct in geheugenplaatsen rijtjes enen en nullen te kunnen opslaan en tevens om geheugenplaatsen te kunnen uitlezen. In feite kunnen we dit met de instructies POKE en PEEK.

POKE: instructie om op een aangegeven geheugenplaats een aangegeven waarde op te slaan.
Vorm:

POKE geheugenplaats, getal

Bijvoorbeeld:

```
POKE 16437,255
```

Het getal 255 wordt dan in een rijtje enen en nullen omgezet en wel volgens de regels van de binaire getallen.

PEEK: instructie om een aangewezen geheugenplaats uit te lezen. Vorm:

PEEK geheugenplaats

Bijvoorbeeld:

```
LET I=PEEK 16437
```

Aan de variabele **I** zal dan de waarde worden toegekend die overeenkomt met het rijtje enen en nullen dat in geheugenplaats 16437 wordt aangetroffen. Deze waarde zal uiteindelijk als een 'gewoon' decimaal getal worden toegekend.

USR: instructie om het BASIC-programma op een aangegeven geheugenplaats te laten vervolgen met een reeks machine-instructies. Nadat de reeks machine-instructies (als subroutine) is afgewerkt wordt het BASIC-programma weer voortgezet.
Vorm:

USR geheugenplaats

We zien in het voorafgaande dat soms een rijtje enen en nullen in een decimaal getal wordt omgezet (PEEK) en soms wordt een decimaal getal in een rijtje enen en nullen omgezet (bij POKE). Dit alles volgens de regels van de binaire getallen. Een binair getal bestaat alleen maar uit de cijfers 0 en 1 en wel volgens de regel, dat de positie van een cijfer met een macht van twee overeenstemt.. De posities worden vanaf de rechterkant geteld en we beginnen steeds met 0 te tellen.
Bijvoorbeeld:

```
1 0 1 0 1 1 0 1
|   |   | |   |
|   |   | |   nulde positie
|   |   | └-tweede positie
|   |   derde positie
|   vijfde positie
zevende positie
```

Dit getal stemt dan overeen met:

1 op nulde positie :	$1 \times 2^0 = 1$
1 op tweede positie:	$1 \times 2^2 = 4$
1 op derde positie :	$1 \times 2^3 = 8$
1 op vijfde positie :	$1 \times 2^5 = 32$

1 op zevende positie: $1 \times 2^7 = 128$

Totaal: $1 + 4 + 8 + 32 + 128 = 173$

De omzetting van een tientallig getal naar een binair getal geschiedt door steeds machten van twee van het getal af te trekken, en wel steeds de grootst mogelijke macht.

Voorbeeld: omzetting 173 naar rijtje enen en nullen:

$$
\begin{array}{rrl}
 & 173 & \\
2^7 = & \underline{128} - & \text{(is de grootst mogelijke macht van 2)} \\
 & 45 & \\
2^5 = & \underline{32} - & \text{(is nu de grootst mogelijke macht)} \\
 & 13 & \\
2^3 = & \underline{8} - & \text{(is nu de grootst mogelijke macht)} \\
 & 5 & \\
2^2 = & \underline{4} - & \text{(is nu de grootst mogelijke macht)} \\
2^0 = & 1 & \text{(is een macht van 2, nl. } 2^0\text{)}
\end{array}
$$

De getallen 2^7, 2^5, 2^3, 2^2 en 2^0 vormen weer het binaire getal: 1 0 1 0 1 1 0 1 (zie voorafgaande).

Appendix 1: Foutboodschappen

Als bij het intypen van een regel een overduidelijke fout wordt gemaakt zal na het intoetsen van NEWLINE een cursor met de letter S rechts naast de plaats waar de fout werd gemaakt, verschijnen. Men moet deze fout eerst herstellen alvorens men de regel bij het programma kan voegen. Als een programma eenmaal is ingetoetst zal het na het commando RUN worden uitgevoerd. Er verschijnt ten slotte altijd een boodschap, bijvoorbeeld:

0/60

Het tweede getal toont het laatste regelnummer dat werd uitgevoerd en het voorste getal heeft steeds een bepaalde betekenis. Zo geldt dat als een programma correct is uitgevoerd, het eerste getal zoals getoond inderdaad gelijk aan 0 is.
In het nu volgende wordt een overzicht van de verschillende boodschappen gegeven, te zamen met de situatie waarbij deze boodschappen kunnen optreden.

CODE	BETEKENIS	SITUATIE WAARBIJ DEZE BOODSCHAP KAN OPTREDEN
0	tijdens het uitvoeren van het programma werd geen fout ontdekt. Deze boodschap treedt ook op als naar een niet bestaand regelnummer wordt gesprongen dat groter is dan elk ander regelnummer	elke, GOTO, GOSUB
1	De tel-variabele is niet in een FOR-instructie aangegeven, maar er is wel een variabele met die naam.	NEXT
2	In een uitdrukking wordt een variabele gebruikt die van te voren nog geen waarde had gekregen. Er wordt een variabele van een array gebruikt zonder dat voor deze variabele door middel van de DIM-instructie ruimte is gereserveerd (en de beginwaarde 0 is gegeven). In een FOR . . . NEXT-instructie waarbij de tel-	elke, DIM, FOR . . . NEXT

CODE	BETEKENIS	SITUATIE WAARBIJ DEZE BOODSCHAP KAN OPTREDEN
	variabele niet van een beginwaarde is voorzien, terwijl de variabele ook niet van te voren een waarde is toegekend.	
3	Index van een variabele van een array stemt niet overeen met de DIM-instructie. Bijv. DIM A(10) staat niet toe dat A(11) gebruikt wordt Als de index negatief is dan wel groter dan 65535, dan verschijnt boodschap B.	variabelen van een array, DIM
4	Bij het standaardgeheugen een veel voorkomende boodschap, nl. 'te weinig geheugenruimte'. Als gevolg van dit gebrek aan geheugenruimte kan het regelnummer na de schuine streep verminkt overkomen.	LET, INPUT, DIM, PRINT, LIST, PLOT, UNPLOT, FOR, GOSUB, en soms tijdens het uitwerken van een functie
5	Het beeldscherm bevat geen ruimte meer voor verdere uitvoer. Druk hierna op CONT; hierdoor zal het beeldscherm gewist worden en het programma wordt voortgezet.	PRINT, LIST
6	Bij berekeningen ontstaan te grote getallen: zgn. 'overflow'. Overflow treedt op als een berekend getal groter is dan ca. 10^{38}. Bijvoorbeeld bij delen door 0.	bij berekeningen
7	Er wordt een RETURN-instructie aangetroffen zonder dat eerst een GOSUB-instructie werd uitgevoerd. Deze foutboodschap treedt o.a. op als de subroutine niet van te voren door een STOP-instructie of op andere wijze van het hoofdprogramma is afgescheiden.	RETURN
8	Er wordt getracht INPUT als commando uit te voeren. INPUT is de enige instructie die niet als commando mag worden uitgevoerd.	INPUT
9	Als resultaat van een STOP-instructie. Als men hierna CONT indrukt zal het programma met de op de STOP-in-	STOP

CODE	BETEKENIS	SITUATIE WAARBIJ DEZE BOODSCHAP KAN OPTREDEN
	structie volgende regel worden voortgezet (als deze regel tenminste bestaat).	
A	Argument van de functie is ongeldig: bijv. bij SQR wordt een negatief getal als argument gegeven.	SQR, LN, ASN, ACS
B	Integer-getal valt buiten toegestane bereik. Er wordt bijvoorbeeld naar een niet bestaand adres gesprongen. Als bij een instructie, commando of functie een integer-waarde vereist is en er wordt een reële waarde opgegeven dan wordt dit getal automatisch eerst afgerond. Als het afgeronde getal buiten het toegestane bereik valt volgt deze foutmelding. De index van een array-variabele is negatief, dan wel groter dan 65535.	RUN, RAND, POKE, DIM, GOTO, GOSUB, LIST, LLIST, PAUSE, PLOT, UNPLOT, CHR$, PEEK, USR, AT
C	De string die door VAL tot een numerieke waarde herleid moet worden kan niet tot een numerieke waarde worden herleid. Bijvoorbeeld: PRINT VAL "12A4".	VAL
D	Bij het onderbreken van een programma met behulp van BREAK.	na het afwerken van een instructie of bij LOAD, SAVE, LPRINT, LLIST of COPY
E	De INPUT-regel begint met STOP	INPUT
F	Bij het opslaan van het programma op cassettetape wordt vergeten om een naam na SAVE op te geven.	SAVE

Indien er problemen zijn . . .

In de praktijk zal het dikwijls voorkomen dat het geheugen vol geraakt (zeker met het standaard aanwezige geheugen).
Er kunnen dan zeer vreemde dingen gebeuren. Zo kan ineens tekst verdwijnen of de computer reageert niet meer op commando's, of . . .
Hier volgen een aantal veel voorkomende situaties.

Tijdens het uitvoeren van informatie verschijnt foutboodschap 5:

Toets CONT in, hierna zal het programma worden voortgezet. Probeer het maar eens met het volgende programma uit:

```
10 FOR K=1 TO 100
20 PRINT K
30 NEXT K
```

Vaak kan men CLS-instructies in het programma opnemen zodat het scherm niet meer vol geraakt.

Tijdens het uitvoeren van een programma verschijnt boodschap 4:

Bekijk de listing (druk eerst op CLEAR) en kijk waar men het programma kan inkorten (kortere namen, kortere regelnummers, REM-instructies weglaten enz.)

Tijdens het intoetsen van een programma gebeuren vreemde zaken:

We doelen hier op het verschijnsel dat er op een gegeven moment tekst verdwijnt en dat de computer niet meer op commando's reageert. Druk zo vaak op de toets RUBOUT tot de cursor weer verschijnt en het programma weer op commando's reageert. Bekijk hierna de listing om te zien hoe een en ander kan worden ingekort.

Het programma reageert vreemd bij een INPUT-instructie:

Toets op STOP (als het om een string gaat wordt eerst RUBOUT ingetoetst). Bekijk hierna de listing om te zien hoe een en ander korter kan worden geschreven, of meer in het algemeen, hoe geheugenruimte gespaard kan worden.

Het programma blijft maar doorrekenen ...

Dit is bijvoorbeeld het geval bij:

```
10 GOTO 20
20 GOTO 10
```

Gebruik in dit geval het commando BREAK. Dit commando kunnen we ook gebruiken als bij het uitlezen van een cassette de computer maar doorgaat met de lees-actie.

Appendix 2: Korte samenvatting van de functies

In het nu volgende geven we een korte samenvatting van de functies die de ZX81 biedt. Merk op dat functies die als resultaat een string opleveren het teken $ achter hun naam hebben staan. Bij wiskundige functies wordt gewerkt met getallen die 9 tot 10 cijfers nauwkeurig zijn waardoor de berekende functie-waarde gewoonlijk tot 8 cijfers nauwkeurig is.

FUNCTIE	ARGUMENT	RESULTAAT
ABS	getal	Geeft de absolute waarde, bijv. ABS(-1) = 1.
ACS	getal	Berekent de arccosinus in radialen. Het argument moet tussen de waarden -1 en 1 liggen.
AND	zijn twee argumenten nodig, rechts dient altijd een getal te staan	De uitdrukking X < 1 AND Y > 2 is alleen waar als beide deeluitdrukkingen waar zijn. Voor getallen: $A \text{ AND } B = \begin{cases} A & \text{als } B \neq 0 \\ 0 & \text{als } B = 0 \end{cases}$ Voor een string: A$ AND B = A$ als B ≠ 0 "" als B = 0 P. M. het teken ≠ wil zeggen 'is ongelijk aan'.
ASN	getal	Berekent de arcsinus in radialen. Het argument moet tussen de waarden -1 en 1 liggen.
AT	twee getallen	Geeft de X- en de Y-coördinaat en bepaalt zodoende waar iets wordt afgedrukt. Bijv. bij PRINT AT 5,5; "*".
ATN	getal	Berekent de arctangens in radialen.
CHR$	getal	Geeft het symbool dat overeenkomt met de code. Als het argument een niet geheel getal is vindt

FUNCTIE	ARGUMENT	RESULTAAT
		eerst afronding plaats.
CODE	string	Geeft de code van het eerste schriftteken van de string. Als de string leeg is wordt de functiewaarde 0.
COS	getal (in radialen)	Berekent de cosinus.
EXP	getal	Berekent de waarde e tot de macht getal.
INKEY$	heeft geen argument	Leest het toetsenbord uit. Het resultaat is de ingetoetste toets (als de cursor de L-mode toont tenminste).
INT	getal	Rond een getal altijd naar beneden af, bijv. INT(3.7) = 3 en INT(-3.7) = -4. Bij INT(A + 0.5) vindt zuivere afronding van A plaats.
LEN	string	Geeft het aantal schrifttekens van de gegeven string, bijv. LEN"AAP" = 3.
LN	getal	Bepaalt de logarithme met het grondtal e. Om de logarithme met het grondtal 10 te vinden moeten we nog door LN(2) delen. Het argument moet groter dan 0 zijn.
NOT	relatie of getal	NOT $X < 2$ is waar als de uitdrukking $X < 2$ onwaar is en andersom als $X < 2$ opgaat is de gehele uitdrukking (dus met NOT) onwaar. NOT met een getal levert 0 als het argument ongelijk aan 0 is, en anders levert het de waarde 1.
OR	twee relaties of twee getallen	De uitdrukking $X < 1$ OR $Y > 2$ is alleen onwaar als beide deeluitdrukkingen onwaar zijn. Voor getallen: $A \text{ OR } B = \begin{cases} 1 & \text{als } B \neq 0 \\ A & \text{als } B = 0 \end{cases}$
PEEK	getal	Levert de inhoud van het door het argument aangegeven geheugen-

FUNCTIE	ARGUMENT	RESULTAAT
		plaats. Het argument wordt, indien nodig, eerst afgerond. Het argument mag niet buiten het interval 0 - 65535 vallen.
PI	-	Levert de waarde π d.w.z. 3.14159265. Op het toetsenbord ziet men het symbool π (onder M-toets) maar op het beeldscherm ziet men PI.
RND	-	Geeft een willekeurig getal ('random getal'). De willekeurige getallen volgen een vaste volgorde. De plaats waar men met uitlezen van willekeurige getallen begint, bepaalt men door van te voren de instructie "RAND-getal" te geven. Als getal = 0 dan bepaalt de tijd dat de TV aanstaat de plaats waar met uitlezen begonnen wordt.
SGN	getal	Geeft het teken van het getal. Bijv. SGN(-3) = -1, SGN(0) = 0, SGN(3) = 1.
SIN	getal (in radialen)	Berekent de sinus.
SQR	getal	Bepaalt wortel van getal, bijv. SQR(9) = 3. Het argument dient groter of gelijk aan 0 te zijn.
STR$	getal	Levert de string die men zou krijgen als het getal (of de uitdrukkking) in een PRINT-instructie zou staan. Bijv. STR$3 geeft "3".
TAB	getal	Bepaalt kolomnummer bij een PRINT-instructie.
TAN	getal (in radialen)	Berekent de tangens.
USR	getal	Roept vanuit het BASIC-programma een in machinecode geschreven programma aan. Het startadres van deze subroutine wordt door het argument van de functie aangegeven. Het argument wordt, indien nodig, eerst afgerond. Het resul-

FUNCTIE	ARGUMENT	RESULTAAT
		taat komt in registerpaar bc te staan. Het argument mag niet buiten het interval 0 - 65535 vallen.
VAL	string	Werkt de string uit alsof het om een numerieke uitdrukking gaat. Bijvoorbeeld VAL"3 + 5"= 8. Als deze VAL-functie zelf in een uitdrukking staat moet deze steeds voorop staan, bijvoorbeeld: LET X = VAL"3 + 5" + 4.

Appendix 3: Korte samenvatting van speciale symbolen

De volgende symbolen worden bij berekeningen gebruikt.

+	Optellen. Dit is het enige bewerkings-symbool dat ook bij strings wordt gebruikt, nl. om strings aan elkaar te rijgen.
-	Aftrekken.
*	Vermenigvuldigen.
/	Delen.
**	Machtsverheffen. (is een afzonderlijke toets voor: H-toets)
( )	Haakjes zoals die gewoonlijk in de algebra worden gebruikt, nl. om bepaalde bewerkingen voorrang te verlenen.
=	Na de term LET dient dit teken als het 'wordt-teken' gelezen te worden: LET N = N + 1 wil zeggen N wordt de oude waarde van N + 1.

Een relatie is een uitdrukking die in een IF ... THEN-instructie wordt gebruikt, bijv. IF X < 3 THEN PRINT X.
Hierin is X < 3 een relatie die afhankelijk van de momentane waarde van X waar dan wel onwaar is. Het teken < brengt X en 3 met elkaar in relatie en wordt daarom een relatie-teken genoemd.
De volgende relatie-tekens mogen worden gebruikt:

=	Geeft een gelijkheid aan.
>	Groter dan.
> =	Groter of gelijk.
<	Kleiner dan.
< =	Kleiner of gelijk.
< >	Ongelijk aan.

Voor al deze tekens bestaat een afzonderlijke toets. In het onderstaande volgt nog een overzicht van de overige symbolen:

⇦	Verplaats de cursor een plaats naar links.
⇩	Verplaats de cursor een plaats naar onderen.

⇧ Verplaats de cursor een plaats naar boven.

⇨ Verplaats de cursor een plaats naar rechts.

"" Ziet men bij de Q-toets: om aanhalingstekens in strings op te nemen.

$ Teken dat altijd na de naam van een string-variabele wordt geplaatst.

" Aanhalingstekens: tussen deze tekens dient men altijd strings te plaatsen, bijv. PRINT "EIND" en LET A$ = "BEGIN".

: Heeft bij de ZX 81 geen speciale betekenis. Bij andere BASIC-versies wordt dit teken soms gebruikt om tekst in een INPUT-instructie onder te brengen. Dit is bij de ZX 81 niet toegestaan.

; Teken dat bij een PRINT-instructie wordt gebruikt om aan te geven dat een volgende zaak direct achter de vorige zaak dient te worden afgedrukt.

? Heeft bij de ZX 81 geen speciale betekenis. Bij andere BASIC-versies wordt dit teken soms gebruikt om aan te geven dat er iets als een gevolg van een INPUT-instructie moet worden ingevoerd.

, Algemeen teken om argumenten te scheiden, bijvoorbeeld in DIM-, PLOT-, UNPLOT- en POKE-instructie. Bij een PRINT-instructie dient het teken om twee af te drukken zaken te scheiden waarbij tijdens het afdrukken automatisch een kolomindeling wordt aangehouden.

. Teken om 'komma' in een getal aan te geven, bijvoorbeeld
3. 7
Overigens heeft het symbool geen speciale betekenis.

Appendix 4: Prioriteitsregels

Prioriteitsregels zijn regels die aangeven hoe een uitdrukking moet worden uitgewerkt. Zo heeft bijvoorbeeld vermenigvuldigen een hogere prioriteit dan optellen en als gevolg hiervan zal bij de uitdrukking

```
LET A=3+4*2
```

aan A de waarde 3 + (4 * 2) = 11 en niet de waarde (3 + 4) * 2 = 14 worden toegekend.
Bij gelijke prioriteit wordt een uitdrukking van links naar rechts uitgewerkt. Zo zal de uitdrukking:

```
LET A=4/2*8
```

de waarde (4/2)* 8 = 16 opleveren en niet 4/(2 * 8) = 0.25.

In het nu volgende ziet men een overzicht van de verschillende bewerkingen en hun prioriteit. De prioriteit wordt door een getal aangegeven: hoe groter het getal hoe hoger de prioriteit. Zo ziet men bijvoorbeeld dat vermenigvuldigen inderdaad een hogere prioriteit heeft dan optellen.

BEWERKING	PRIORITEITS-CIJFER
bepaling van index van variabele van een array	12
bepaling van een substring	12
berekenen van functie-waarden (een uitzondering hierop vormt de functie NOT)	11
machtsverheffen: **	10
betrekken van een min-teken bij een getal, in een uitdrukking waarbij dit min-teken niet tussen twee getallen staat. Bijvoorbeeld: LET A=-2	9
vermenigvuldigen en delen: * en /	8
optellen en aftrekken: + en -	6
=, > , < , < =, > =, <>	5
NOT	4

BEWERKING	PRIORITEITS-CIJFER
AND	3
OR	2

Door haakjes te gebruiken kan men deze regels negeren.

Appendix 5: Instructies en commando's

In het nu volgende wordt een samenvatting van verschillende instructies en commando's gegeven. Op INPUT na kunnen alle instructies ook als commando's worden gegeven. Het meest interessant in dit verband is de PRINT-instructie, bijvoorbeeld:

```
PRINT 25*35
```

Men ziet dan direct het antwoord: 875. In een dergelijk geval spreekt men van een directe PRINT-instructie. Commando's kunnen ook in programma's worden opgenomen. Als zodanig is er bij de ZX 81 in feite geen onderscheid tussen termen die gewoonlijk voor instructies worden gebruikt en termen die gewoonlijk voor commando's worden gebruikt. Voert men een commando uit dan wordt het scherm eerst gewist (behalve bij: COPY).

Bij de notatie zullen we de notatie van de officiële ZX 81 handleiding overnemen.
Dit houdt het volgende in:

SYMBOOL	BETEKENIS
α	representeert een enkele letter
v	stelt een variabele voor
x, y en z	stellen numerieke uitdrukkingen voor (expressies)
m en n	stellen numerieke uitdrukkingen voor, die altijd worden afgerond
e	stelt een algemene uitdrukking voor (kan o. a. ook op strings betrekking hebben)
f	stelt een specifieke uitdrukking met strings voor
s	stelt een instructie voor

Hieronder volgen de verschillende instructies/commando's:

UITDRUKKING	BETEKENIS
CLEAR	Wist alle toegekende waarden van variabelen. Op deze wijze wordt geheugenruimte vrijgemaakt. Wil men fouten in een programma d. m. v. PRINT-

UITDRUKKING	BETEKENIS
	opdrachten opsporen dan is het verstandig om voor het runnen steeds het commando CLEAR te geven.
CLS	Maakt het beeldscherm schoon. Wat op het beeldscherm staat afgebeeld noemt men de display-file. In feite is dit een gedeelte van het geheugen.
CONT	Indien bij het stoppen van een programma de melding p/q volgt (bijv. : 9/40 dus p = 9 en q = 40) dan heeft het intoetsen van CONT het volgende effect: als p ongelijk aan 9 is zal het programma op regelnummer q worden voortgezet en anders wordt het programma op de daarop volgende regel voortgezet. CONT wordt met name gebruikt om het programma voort te zetten als het scherm vol is.
COPY	Heeft tot gevolg dat de afbeelding op het beeldscherm op de printer wordt gekopieerd.
DIM $\alpha(n_1, \ldots, n_k)$	Als gevolg van deze instructie wordt ruimte gecreeerd voor n_1 x n_2 x ... n_k variabelen. Deze variabelen kan men daarna in het programma aanroepen met de hoofdnaam α en de indices n_1 t/m n_k. Voorbeeld: DIM A(10) creëert ruimte voor de variabelen A(1) t/m A(10). Deze variabelen kunnen als iedere andere variabele worden gebruikt, bijvoorbeeld: LET A(3) = 4 + A(1). De verzameling variabelen die op deze wijze wordt geïntroduceerd noemt men de array. De beginwaarde van iedere variabele is 0. Een reeds bestaande array met hoofdnaam α wordt als gevolg van de DIM-instructie met dezelfde hoofdnaam gewist. Als er niet genoeg ruimte in het geheugen is verschijnt foutboodschap nr. 4. Variabelen van een array moeten in tegenstelling tot sommige andere BASIC-versies bij de ZX 81 altijd met behulp van de DIM-instructie geïntroduceerd worden.
DIM $\alpha\$(n_1, \ldots, n_k)$	Als het voorafgaande, maar nu gaat het om stringvariabelen. Er is echter een verschil: de laatste index geeft wat de vaste lengte van de string is. Zo introduceert DIM A$(5. 10) in feite 5 string-variabelen die strings van 10 schrifttekens kunnen bevatten, bijvoorbeeld: LET A$(3) = "A1B4C6D8E0".
FAST	Heeft tot gevolg dat de computer in de zgn. snelle modus gaat rekenen. Het beeldscherm licht alleen

UITDRUKKING	BETEKENIS
	op: aan het einde van een programma, bij een INPUT-instructie en tijdens een PAUSE-instructie (alleen voor ZX 81).
FOR α = x TO y STEP z (NEXT α)	Als gevolg van deze instructie zullen alle instructies tussen de regel met FOR α = x etc. en NEXT α voor de waarden α, α + z, α + 2z, α + 3z etc. herhaaldelijk worden ingevoerd en wel tot de opgehoogde waarde nog kleiner of gelijk is aan y. De variabele α wordt de tel-variabele genoemd.

FOR α = x TO y (NEXT α)	Als boven, maar nu wordt de tel-variabele automatisch steeds met 1 opgehoogd. Zo zullen bij het kleine programma:
	`10 FOR K=1 TO 10`
	`20 PRINT K`
	`30 NEXT K`
	de getallen 1 t/m 10 onder elkaar verschijnen.

GOSUB n (RETURN)	Het programma wordt voortgezet op het aangegeven regelnummer. Nadat de computer de instructie RETURN tegenkomt, wordt het programma weer voortgezet vanaf de regel die volgt op de GOSUB-instructie. Het deel-programma dat door middel van deze tijdelijke sprong is uitgevoerd noemt men de subroutine. Merk op dat n ook een numerieke uitdrukking in plaats van een getal mag zijn.
GOTO n	Zet het programma voort op regelnummer n of indien n niet bestaat op het regelnummer dat het eerst op n volgt. Merk op dat n hier een numerieke uitdrukking mag zijn in plaats van alleen maar een getal. Door middel van deze mogelijkheid kan men de ON ... GOTO-instructie die bij de ZX 81 niet is toegestaan eenvoudig nabootsen.
IF x THEN s	Indien de uitdrukking x waar is wordt de instructie na THEN uitgevoerd. Als de uitdrukking niet waar is wordt het programma met de volgende regel voortgezet. Voorbeeld: IF x < 0 THEN GOTO 100. Als x kleiner is dan 0 wordt het programma bij regel 100 voortgezet.
INPUT v	Instructie voor het invoeren van informatie. Bij een

UITDRUKKING	BETEKENIS
	getal verschijnt de cursor met de letter L. Als een string moet worden ingevoerd verschijnen aanhalingstekens. Deze instructie zorgt voor een tijdelijke onderbreking. Pas na het intypen van de waarde voor de variabele en het intoetsen van NEWLINE wordt het programma voortgezet. Bijvoorbeeld INPUT A (voor invoer getal) en INPUT B$ (voor invoer van een string). In plaats van een getal alleen (of een enkele string) mag men ook een uitdrukking intoetsen. Deze wordt dan automatisch eerst in een getal (of string) omgezet. Met het invoeren van STOP wordt het programma onderbroken (bij het invoeren van STOP als een stringvariabele verwacht wordt, eerst RUBOUT intoetsen).
LET v = e	Instructie om aan de variabele v de waarde van de uitdrukking e toe te kennen. Het teken = dient hier als wordt-teken gelezen te worden. Zo wil LET A = 3 zeggen: A wordt 3; en LET A = A + 3 wil zeggen A krijgt de waarde die A had +3. Het woord LET mag niet worden weggelaten. Als het om een string-variabele gaat met vaste lengte (bij array of bij 'slicing') wordt e afgekapt of met spaties opgevuld.
LIST	Zie onderstaande: LIST 0.
LIST N	Toont het programma op het beeldscherm vanaf regel N. Als het programma te lang is kunnen er twee foutboodschappen optreden. Als het geheugen vol is ziet men de code 4 en als het scherm vol is (maar het geheugen is nog niet vol) ziet men de boodschap 5. Men drukt hierna op CONT.
LLIST	Als LIST 0 maar nu wordt de printer in plaats van het beeldscherm gebruikt. Doorgaans zullen er geen problemen optreden als de printer niet is aangesloten. Bij problemen kan men het programma met BREAK onderbreken.
LLIST N	Als LIST N maar nu wordt de printer in plaats van het beeldscherm gebruikt. Doorgaans zullen er geen problemen optreden als de printer niet is aangesloten. Bij problemen kan men het programma met BREAK onderbreken.
LOAD f	Voor het uitlezen van een programma (met naam f)

UITDRUKKING	BETEKENIS
	van cassette naar geheugen. Als f ontbreekt wordt het eerst voorkomende programma op de cassette ingelezen. Gaat er iets mis: onderbreek dan met BREAK, wis geheugen met NEW, stel knoppen van cassetterecorder opnieuw in en start LOAD-actie weer.
LPRINT	Als PRINT, maar nu wordt de printer in plaats van het beeldscherm gebruikt. Een regel wordt afgedrukt als: - van een regel naar een volgende gesprongen wordt - na een LPRINT-instructie die niet met , of ; wordt afgesloten - als ten gevolge van een TAB-instructie naar een nieuwe regel gesprongen moet worden - aan het eind van het programma. Bij AT wordt het regelnummer genegeerd.
NEW	Het geheugen wordt gewist om een nieuw BASIC-programma te kunnen invoeren.
NEXT α	Zie FOR α = x TO y STEP z.
PAUSE n	Last een pauze in ter grootte van n tijdseenheden (voor n = 50 heeft men een wachttijd van ca. 1 seconde). De pauze kan onderbroken worden door op een toets te drukken. Bij de ZX 80 dient men na deze instructie steeds de instructie POKE 16437,255 te plaatsen.
PLOT m,n	met behulp van de PLOT-functie kunnen bepaalde plaatsen zwart worden gemaakt. Daartoe is het beeldscherm in 0 - 63 horizontale en 0 - 43 verticale posities verdeeld. Bij PLOT m,n zal het beeldpunt met coördinaten m en n zwart worden gemaakt. Hiermee is tevens de print-positie naar de aangegeven plaats verplaatst.
POKE m,n	Sla getal n op in geheugenplaats m. m mag de waarden 0 t/m 65535 aannemen en n de waarden 0 t/m 255. Wordt met name gebruikt om machine-instructies op te nemen.
PRINT ...	Algemene instructie om iets op het beeldscherm te tonen. ; als laatste teken wil zeggen: druk volgende direct achter voorafgaande af. , als laatste teken wil zeggen: druk volgende in

UITDRUKKING	BETEKENIS
	volgende kolom af. Af te drukken zaken steeds met , scheiden. Strings tussen tekens " plaatsen. Als uitkomst te groot of te klein is om geheel in cijfers te verschijnen, vindt automatisch omzetting naar wetenschappelijke notatie plaats. PRINT zonder meer heeft het effect van een regel overslaan. De print-positie kan men regelen door: - de TAB-functie (zie TAB) - de AT-functie (zie AT) - de PLOT-instructie (zie PLOT) Als het scherm vol is verschijnt foutmelding 5 en we kunnen het programma met behulp van het CONT-commando voortzetten.
RAND	Komt overeen met RAND 0 (zie RAND n).
RAND n	Instructie om aan te geven waar met het trekken van random-getallen begonnen moet worden. Wordt altijd in combinatie met RND-functie gebruikt (zie RND bij appendix 2).
REM	Instructie om commentaar in een programma op te nemen, dat dan tijdens het executeren van het programma niet op het beeldscherm verschijnt. Vorm: regelnummer REM commentaar. Iedere nieuwe regel commentaar dient weer met een hoger regelnummer en REM te beginnen.
RETURN	Instructie die bij een subroutine wordt gebruikt om terugkeer naar het hoofdprogramma aan te geven (zie GOSUB).
RUN	Komt overeen met RUN 0 (zie RUN n).
RUN n	Wis alle variabelen en start met het uitvoeren van het programma vanaf het aangegeven regelnummer (indien n niet voorkomt ; met het eerstvolgende regelnummer).
SAVE f	Instructie om aan te geven dat een in het geheugen opgeslagen programma op cassette geschreven dient te worden. Met f kunnen we een naam aan het programma toekennen.

UITDRUKKING	BETEKENIS
SCROLL	Verplaats alle regels op het beeldscherm een positie hoger, zodat de bovenste regel wegvalt en onder een lege regel verschijnt.
SLOW	De computer gaat met een langzamere snelheid rekenen waardoor het beeldscherm voortdurend blijft oplichten (geldt alleen voor ZX 81).
STOP	Als gevolg van deze instructie zal het uitvoeren van het programma stoppen (foutmelding 9). Met CONT kunnen we het programma eventueel weer voortzetten.
UNPLOT m,n	Zie PLOT m,n: als een beeldpunt met coördinaten m en n zwart is zal als gevolg van UNPLOT m,n deze zwarte stip weer worden gewist.

Appendix 6: Symbolen, schrifttekens en codes

In het nu volgende ziet men een volledig overzicht van alle symbolen, schrifttekens en bijbehorende codes. Merk op dat bij een aantal codes geen enkel symbool staat, dat wil zeggen deze codes worden in feite niet gebruikt.
In de oorspronkelijke Engelse handleiding staan ook codes die te maken hebben met assembler-programmering. Omdat assembler-programmering niet tot het doel van dit boek behoort zijn deze gegevens hier met opzet weggelaten.

Code	Symbool	Code	Symbool	Code	Symbool
0	spatie	28	0	56	S
1	▘	29	1	57	T
2	▝	30	2	58	U
3	▀	31	3	59	V
4	▖	32	4	60	W
5	▌	33	5	61	X
6	▞	34	6	62	Y
7	▛	35	7	63	Z
8	▒	36	8	64	RND
9	🮏	37	9	65	INKEY$
10	🮎	38	A	66	PI
11	"	39	B	67	
12	£	40	C	68	
13	$	41	D	69	
14	:	42	E	70	
15	?	43	F	71	
16	(	44	G	72	
17	)	45	H	73	
18	>	46	I	74	
19	<	47	J	75	wordt niet gebruikt
20	=	48	K	76	
21	+	49	L	77	
22	-	50	M	78	
23	*	51	N	79	
24	/	52	O	80	
25	;	53	P	81	
26	,	54	Q	82	
27	.	55	R	83	

Code	Symbool	Code	Symbool	Code	Symbool
84		126	number	168	inverse C
85		127	cursor	169	inverse D
86		128	█	170	inverse E
87		129	▟	171	inverse F
88		130	▙	172	inverse G
89		131	▄	173	inverse H
90		132	▜	174	inverse I
91		133	▐	175	inverse J
92		134	▚	176	inverse K
93		135	▗	177	inverse L
94		136	▒	178	inverse M
95		137	🮑	179	inverse N
96		138	🮒	180	inverse O
97		139	inverse " *	181	inverse P
98	wordt niet gebruikt	140	inverse £	182	inverse Q
99		141	inverse $	183	inverse R
100		142	inverse :	184	inverse S
101		143	inverse ?	185	inverse T
102		144	inverse (	186	inverse U
103		145	inverse)	187	inverse V
104		146	inverse >	188	inverse W
105		147	inverse <	189	inverse X
106		148	inverse =	190	inverse Y
107		149	inverse +	191	inverse Z
108		150	inverse -	192	" "
109		151	inverse *	193	AT
110		152	inverse /	194	TAB
111		153	inverse ;	195	niet gebruikt
112	cursor ⇧	154	inverse ,	196	CODE
113	cursor ⇩	155	inverse .	197	VAL
114	cursor ⇦	156	inverse 0	198	LEN
115	cursor ⇨	157	inverse 1	199	SIN
116	GRAPHICS	158	inverse 2	200	COS
117	EDIT	159	inverse 3	201	TAN
118	NEWLINE	160	inverse 4	202	ASN
119	RUBOUT	161	inverse 5	203	ACS
120	K or L mode	162	inverse 6	204	ATN
121	FUNCTION	163	inverse 7	205	LN
122		164	inverse 8	206	EXP
123	wordt niet gebruikt	165	inverse 9	207	INT
124		166	inverse A	208	SQR
125		167	inverse B	209	SGN

* N. B. de term 'inverse' geeft aan dat het symbool nu in wit tegen een zwarte achtergrond wordt afgebeeld.

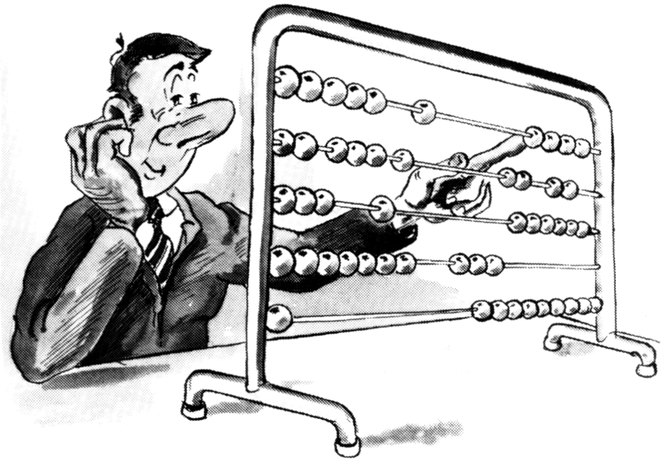
Code	Symbol	Code	Symbol	Code	Symbol
210	ABS	225	LPRINT	240	LIST
211	PEEK	226	LLIST	241	LET
212	USR	227	STOP	242	PAUSE
213	STR$	228	SLOW	243	NEXT
214	CHR$	229	FAST	244	POKE
215	NOT	230	NEW	245	PRINT
216	**	231	SCROLL	246	PLOT
217	OR	232	CONT	247	RUN
218	AND	233	DIM	248	SAVE
219	< =	234	REM	249	RAND
220	> =	235	FOR	250	IF
221	< >	236	GOTO	251	CLS
222	THEN	237	GOSUB	252	UNPLOT
223	TO	238	INPUT	253	CLEAR
224	STEP	239	LOAD	254	RETURN
				255	COPY

Programma's

In het nu volgende zijn een aantal programma's opgenomen: spelletjes en serieuze toepassingen. Spelletjes zijn niet alleen boeiend, maar ook uiterst leerzaam. Probeer daarom maar eens na te gaan hoe een en ander in BASIC is verwoord. De toepassingen zijn voor een groot deel gericht op de statistiek.
Bij de programma's maken we nog de volgende opmerkingen:

- Alle programma's zijn uitgetest. Vaak zijn de regelnummers met 2 opgehoogd om geheugen te sparen.
  Een kleine wijziging kan bij sommige programma's al fataal zijn. Alle programma's kunnen met het standaard 1 K geheugen worden gebruikt.
- Pas bij het intoetsen vooral op bij de tekens **, < > , < = en > =. Deze tekens moet men steeds door één toets invoeren.
- Bij PAUSE-instructies is steeds de instructie POKE 16437,255 opgenomen. Deze instructie geldt alleen voor de ZX80.
- In de meeste gevallen kan men controleren of men het programma correct heeft ingevoerd door naar het voorbeeld te kijken. Daar waar uitkomsten door RND-functies worden bepaald kunnen verschillen optreden.
- De lege string "" verkrijgt men door tweemaal achter elkaar " in te toetsen en niet door op de Q-toets te drukken.
- Verantwoording:
  Het programma EEUWKALENDER is gebaseerd op een idee dat in een handleiding voor HP-calculators (HP-45 applications book) werd aangetroffen, en het programma MAANLANDER is gebaseerd op een idee dat in een handleiding van de Texas Instruments SR52 werd beschreven.
- De overige programma's zijn, dan wel op de gebruikelijke standaardformules gebaseerd dan wel zijn ze voor zover bekend origineel.
- Bij de voorbeelden is onderstreept wat door de gebruiker is ingevoerd.

Rekenoefeningen



Dit is een aardig programma om jonge kinderen reken-opgaven te geven. We kunnen kiezen tussen optellen, aftrekken en vermenigvuldigen en wel door onmiddellijk een +, - of * in te toetsen. Hierna worden 10 eenvoudige sommen getoond. Ieder getal bestaat slechts uit één cijfer. Als bij een bepaalde opgave tweemaal een fout antwoord gegeven is, dan toont de computer automatisch het juiste antwoord. Na de 10 opgaven wordt als score het aantal gemaakte fouten getoond. Een score gelijk aan 0 wil dus zeggen: geen enkele fout gemaakt!

Programma

```
 2 PRINT "KIES +,-, OF*"
 4 INPUT K$
 6 LET X=0
 8 FOR K=1 TO 10
10 PAUSE 75
12 POKE 16437,255                (alleen voor ZX80)
14 CLS
16 LET F=0
18 PRINT "SOM ";K
```

```
20 LET A=INT(RND*9+1.)
22 LET B=INT(RND*9+1.)
24 IF B>A THEN GOSUB 60
26 LET CK=CODE K$
28 IF CK=21 THEN LET C=A+B
30 IF CK=22 THEN LET C=A-B
32 IF CK=23 THEN LET C=A*B
34 PRINT A;CHR$ CK;B;"="
36 PRINT "VOER GETAL IN"
38 INPUT AN
40 LET F=F+1
42 IF C-AN=0 THEN LET G$="GOED"
44 IF C-AN<>0 THEN LET G$="FOUT"
46 IF C-AN<>0 THEN LET X=X+1
48 PRINT AN;" IS ";G$
50 IF F=2 THEN PRINT "ANTWOORD=";C
52 IF F=2 THEN GOTO 56
54 IF C-AN<>0 THEN GOTO 36
56 NEXT K
58 PRINT "SCORE=";X
60 LET H=A
62 LET A=B
64 LET B=H
66 RETURN
```

<u>Voorbeeld</u>

KIES +, -, OF *
<u>+</u>
SOM 1
7 + 3 =
VOER GETAL IN
<u>10</u>
10 IS GOED
SOM 2
7 + 5 =
VOER GETAL IN etc.

Mastermind

Als er een spel de laatste jaren populair geworden is, dan is het wel MASTERMIND. De charme van het spel ligt ongetwijfeld in zijn eenvoud.
Oorspronkelijk wordt het spel met gekleurde pennetjes gespeeld. Bij de computer maken we gebruik van cijfers. De computer heeft 4 cijfers onzichtbaar in 4 vakjes gestopt en u dient die cijfers te raden door steeds 4 cijfers op te noemen. De computer meldt dan hoeveel cijfers van het door u genoemde rijtje inderdaad in de vakjes liggen en bovendien hoeveel al in de juiste volgorde staan. Voor alle duidelijkheid: als de computer de rij 1 5 4 7 in 'gedachten' heeft en u noemt 4 5 3 8 dan meldt de computer "2 GETALLEN JUIST" (nl. 4 en 5) en "1 OP JUISTE PLAATS" (nl. 5).
Mocht u het oorspronkelijke spel nog niet hebben dan zal de ervaring met onderstaand programma u ongetwijfeld naar de winkel doen rennen. Om het u wat gemakkelijker te maken vermelden we alvast dat de computer steeds 4 verschillende cijfers in gedachten heeft.
Veel plezier met MASTERMIND!

Programma

```
 5 DIM A(4)
 7 DIM B(4)
 9 RAND 0
11 LET N=0
13 FOR K=1 TO 4
15 LET A(K)=INT(RND*9+1.)
17 NEXT K
19 FOR K=1 TO 3
21 FOR J=K+1 TO 4
23 IF A(K)=A(J) THEN GOTO 13
25 NEXT J
27 NEXT K
29 FOR K=1 TO 4
```

```
31 PRINT "GETAL";K;"="
33 INPUT B(K)
35 PRINT B(K)
37 NEXT K
39 LET N=N+1
41 LET G=0
43 LET P=0
45 FOR K=1 TO 4
47 FOR J=1 TO 4
49 IF A(K)=B(J) THEN LET G=G+1
51 NEXT J
53 IF A(K)=B(K) THEN LET P=P+1
55 NEXT K
57 PRINT G;" GETALLEN JUIST"
59 PRINT P;" OP JUISTE PLAATS"
61 PAUSE 150
63 POKE 16437,255                 (alleen voor ZX80)
65 CLS
67 IF P=4 THEN GOTO 71
69 GOTO 29
71 PRINT "OK SCORE=";N
```

Voorbeeld

GETAL 1 =
3
GETAL 2 =
5
GETAL 3 =
1
GETAL 4 =
7
2 GETALLEN JUIST
0 OP JUISTE PLAATS
GETAL 1 =
etc.

Sparen 1

Onderstaand programma toont het gespaarde bedrag als men iedere termijn een bepaald bedrag stort. Uiteraard wordt het eindbedrag bepaald door de rente en het aantal keren dat men het bedrag stort.

Programma

```
10 PRINT "SPAREN"
20 PRINT "RENTE="
30 INPUT J
40 PRINT J
50 LET J=J*0.01
60 PRINT "BEDRAG PER TERMYN="
70 INPUT PMT
80 PRINT PMT
90 PRINT "AANTAL TERMYNEN="
100 INPUT N
110 PRINT N
120 LET TEL=((J+1)**N-1)*PMT
```

```
130 PRINT "TOEKOMSTIGE WAARDE=";TEL/J
```

Voorbeeld

SPAREN
RENTE =
8
BEDRAG PER TERMYN =
1000
AANTAL TERMYNEN =
5
TOEKOMSTIGE WAARDE = 5866.601

N. B. Het getoonde voorbeeld toont dat als u 5 jaar achter elkaar *f* 1000,- naar de bank brengt (met 8% rente) u uiteindelijk *f* 5866,60 heeft gespaard.
Op regel 120 ziet men het teken ** , dit kan men met behulp van één toets intoetsen,nl. met de H-toets.

Sparen 2

Dit programma sluit in feite aan bij SPAREN 1. Nu stellen we ons de vraag hoe een bepaald eindbedrag verkregen wordt als we de rente en het aantal termijnen kennen.

Programma

```
10 PRINT "BETALING PER TERMYN"
20 PRINT "RENTE="
30 INPUT J
40 PRINT J
50 PRINT "TOEKOMSTIGE WAARDE="
60 INPUT FV
70 PRINT FV
80 PRINT "AANTAL TERMYNEN="
90 INPUT N
100 PRINT N
110 LET NOEM=FV*J*0.01
120 LET TEL=(1+0.01*J)**N-1
130 LET PMT=NOEM/TEL
140 PRINT "PER TERMYN TE BETALEN=";PMT
```

Voorbeeld

BETALING PER TERMYN
RENTE =
8
TOEKOMSTIGE WAARDE =
10000
AANTAL TERMYNEN =
5
PER TERMYN TE BETALEN = 1704,56

Gemiddelde en standaarddeviatie

Gemiddelde en standaarddeviatie vormen twee pijlers in de statistiek. Het gemiddelde wordt bepaald door de som van de getallen te nemen en deze waarde door het aantal getallen (N) te delen. Bij de standaarddeviatie gaan we uit van het verschil tussen iedere waarde en het gemiddelde.
Al deze verschillen worden gekwadrateerd en daarna opgeteld. Deze waarde wordt door N-1 gedeeld en de wortel van het zo verkregen resultaat levert ons uiteindelijk de standaarddeviatie.
Bovenstaande zinnen worden in feite door het programma op meer beknopte wijze beschreven. Het programma vraagt eerst het aantal getallen op en vervolgens wordt ieder getal afzonderlijk ingevoerd.

Programma

```
10 PRINT "AANTAL="
20 INPUT N
30 LET S=0
40 LET S2=0
50 FOR K=1 TO N
60 PRINT "VOER GETAL ";K;" IN"
70 INPUT X
80 CLS
90 LET S=S+X
100 LET S2=S2+X*X
110 NEXT K
120 PRINT "GEMIDD.=";S/N
130 LET V=(S2-S*S/N)/(N-1)
140 PRINT "S.DEV=";SQR(V)
```

Voorbeeld

AANTAL =
4
VOER GETAL 1 IN

```
4
VOER GETAL 2 IN
4
VOER GETAL 3 IN
5
VOER GETAL 4 IN
4
GEMIDD. = 4.25
S. DEV = 0.5
```

Regressie en correlatie

De regressie- en correlatieberekening houdt zich bezig met de vraag hoe men door een gegeven aantal punten zo goed mogelijk een rechte lijn kan trekken. Deze rechte lijn wordt door twee grootheden getypeerd, nl. de hellingshoek M en het punt waar de lijn de Y-as doorsnijdt (B). Ieder punt wordt getypeerd door een X- en een Y-coördinaat. Het programma vraagt allereerst hoeveel punten er worden ingevoerd.
Na het invoeren van de coördinaten worden dan de waarden M en B berekend. Ten slotte wordt de correlatie-coëfficiënt R getoond. Als alle punten precies op een rechte lijn liggen zal de absolute waarde van R gelijk aan 1 zijn. Indien de punten niet precies op een rechte liggen zal de absolute waarde van R minder dan 1 zijn. Kennelijk geeft de waarde van R aan, hoe goed de punten wel op de berekende rechte lijn liggen.

Programma

```
 5 PRINT "N="
10 INPUT N
15 PRINT N
20 LET XS=0
25 LET YS=0
30 LET X2=0
35 LET Y2=0
40 LET XY=0
45 FOR K=1 TO N
50 PRINT "X";K;"="
55 INPUT X
60 PRINT X
65 PRINT "Y";K;"="
70 INPUT Y
75 PRINT Y
```

```
80 PAUSE 75
85 POKE 16437,255                    (alleen voor ZX80)
90 CLS
95 LET XS=XS+X
100 LET YS=YS+Y
105 LET X2=X2+X*X
110 LET Y2=Y2+Y*Y
115 LET XY=XY+X*Y
120 NEXT K
125 LET M=(XS*YS/N-XY)/(XS*XS/N-X2)
130 PRINT "M=";M
135 PRINT "B=";YS/N-M*XS/N
140 LET NM=SQR((N*X2-XS*XS)*(N*Y2-YS*YS))
145 LET T=N*XY-XS*YS
150 PRINT "R=";T/NM
```

Voorbeeld

N =
2
2
X1 =
1
Y1 =
2
X2 =
2
Y2 =
4
M = 2
B = 0
R = 1

Faculteit

Onder de faculteit van een getal verstaan we de waarde die we krijgen door de volgende handeling uit te voeren: getal * (getal - 1) * (getal - 2)... * 2 * 1 = .
Bijvoorbeeld, de faculteit van 3 is 3*2*1 = 6.
Men noteert de faculteit van 3 als 3! De faculteit geeft aan op hoeveel manieren we een aantal voorwerpen kunnen rangschikken. Zo kunnen 3 voorwerpen op 6 manieren gerangschikt worden. Het programma gaat uit van gehele (positieve) getallen.

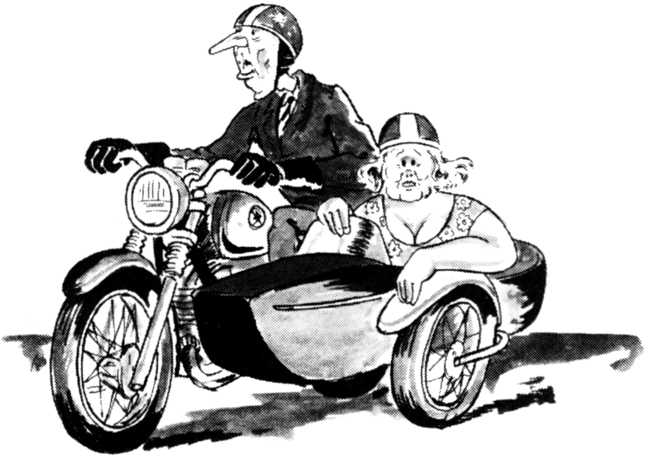
Programma

```
10 PRINT "VOER N IN"
20 INPUT N
30 LET F=N
40 IF N=1 OR N=0 THEN LET F=1
50 IF N=1 OR N=0 THEN GOTO 90
60 LET F=F*(N-1)
70 LET N=N-1
80 IF N>1 THEN GOTO 60
90 PRINT "FACULTEIT=";F
```

Voorbeeld

VOER N IN
3
FACULTEIT = 6

Combinaties



Stel dat we n verschillende voorwerpen hebben (n is bijvoorbeeld 7) dan kunnen we ons de vraag stellen op hoeveel manieren we hier rijtjes van m voorwerpen (m is bijvoorbeeld 5) uit kunnen pakken. Dit aantal wordt aangeduid met de notatie $\binom{n}{m}$ (uitspraak:'n over m') en dit aantal is gelijk aan de faculteit van n, gedeeld door de faculteit van m maal de faculteit van n - m. Deze waarden worden ook wel eens als binomiaal-coëfficiënten aangeduid (zie ook PERMUTATIES).

Programma

```
10 PRINT "N="
20 INPUT N
30 PRINT "M="
40 INPUT M
50 LET F=N
60 GOSUB 300
70 LET NA=FA
80 LET F=M
90 GOSUB 300
100 LET T1=FA
```

```
110 LET F=N-M
120 GOSUB 300
130 LET T2=FA
140 PRINT "C=";NA/(T1*T2)
150 STOP
300 IF F=0 OR F=1 THEN LET FA=1
310 IF F=0 OR F=1 THEN GOTO 360
320 LET FA=F
330 LET FA=FA*(F-1)
340 LET F=F-1
350 IF F>1 THEN GOTO 330
360 RETURN
```

Voorbeeld

N =
7
M =
5
C = 21

Permutaties

Stel dat we n verschillende voorwerpen hebben (n is bijvoorbeeld 8) dan kunnen we ons de vraag stellen op hoeveel manieren we hier rijtjes van m voorwerpen (m is bijvoorbeeld 4) uit kunnen samenstellen waarbij rekening wordt gehouden met de volgorde (zo is de reeks a b c d bijvoorbeeld verschillend van de reeks a c b d: volgorde is anders). Merk het verschil met COMBINATIES op; hier werd niet met de volgorde rekening gehouden. Om een en ander wat aanschouwelijk te maken het volgende. Stel dat we 13 speelkaarten hebben (harten-2 t/m harten-aas) dan kunnen we de vraag stellen, hoeveel verschillende mogelijkheden er zijn als we 7 kaarten in de hand hebben. In dit geval gaat het om combinaties omdat de volgorde hierbij niet van belang is. Stellen we ons de vraag hoeveel mogelijkheden er zijn als we de volgorde waarin de 7 kaarten zijn gegeven wel in ogenschouw nemen, dan hebben we te maken met permutaties.
Het aantal permutaties wordt berekend door de faculteit van n te delen door de faculteit van n - m.

Programma

```
10 PRINT "N="
20 INPUT N
30 PRINT "M="
40 INPUT M
50 LET F=N
60 GOSUB 300
70 LET NA=FA
80 LET F=N-M
90 GOSUB 300
100 LET T=FA
110 PRINT "P=";NA/T
120 STOP
300 IF F=0 OR F=1 THEN LET FA=1
310 IF F=0 OR F=1 THEN GOTO 360
```

```
320 LET FA=F
330 LET FA=FA*(F-1)
340 LET F=F-1
350 IF F>1 THEN GOTO 330
360 RETURN
```

Voorbeeld

N =
8
M =
4
P = 1680

Opmerking

Indien men vaak zowel combinaties en permutaties moet berekenen, dan kan men programma COMBINATIES eenvoudig uitbreiden met de regel:

145 PRINT "P="; NA/T2

Normaal-verdeling

In de statistiek speelt de normaal-verdeling (Gauss-verdeling) een zeer voorname rol. Het nu volgende programma berekent voor een willekeurige waarde X de kans dat een waarneming, kleiner dan X, gevonden wordt en wel als de uitkomsten standaardnormaal verdeeld zijn. Dat wil zeggen: de normaal-verdeling heeft een theoretisch gemiddelde 0 en een standaarddeviatie 1. Op regel 50 zien we de uitdrukking PI. Deze uitdrukking correspondeert met het symbool π onder de toets M.

Programma

```
10 PRINT "X="
20 INPUT X
30 PRINT X
40 LET T=1/(1+0.2316419*X)
50 LET C=1/SQR(2*PI)
60 LET FD=EXP(-X*X/2)*C
70 LET A1=0.3193815*T
80 LET A2=-0.356563782*T*T
90 LET A3=1.781477937*T*T*T
100 LET A4=-1.821255978*T*T*T*T
110 LET A5=1.330274429*T*T*T*T*T
120 LET F=1-FD*(A1+A2+A3+A4+A5)
130 PRINT "F=";F
```

Voorbeeld

X =
1.645
F = 0.95001511

Poisson-verdeling

De poisson-verdeling is gekoppeld aan de zgn. statistiek van de zeldzame gebeurtenissen. Kansen bijvoorbeeld met betrekking tot blikseminslagen, verkeersongelukken enz. zijn gewoonlijk gekoppeld aan deze verdeling.
Er worden maar twee grootheden opgevraagd, nl. L (stelt gemiddelde waarde voor) en K (stelt het aantal keren voor waarvoor de kans wordt uitgerekend).
Om een en ander meer tastbaar te maken het volgende. Stel dat er in de laatste 50 jaar in een bepaald gebied (en in een bepaalde maand, bijv. mei) 85 blikseminslagen zijn geweest, dan komt dit neer op een gemiddelde L van 85/50 = 1.7 blikseminslagen/maand mei.
We kunnen dan met het programma berekenen wat de kans is dat er in de komende maand mei precies 6 blikseminslagen zullen zijn. Uit het voorbeeld na het programma kan men afleiden dat deze kans P gelijk is aan 0.006 ...

<u>Programma</u>

```
10 PRINT "L="
20 INPUT L
30 PRINT "K="
40 INPUT K
50 LET T1=EXP(-L)
60 LET T2=L**K
70 LET F=K
80 GOSUB 300
90 LET N=FA
100 LET P=T1*T2/N
110 PRINT "L=";L
120 PRINT "K=";K
130 PRINT "P=";P
140 STOP
300 IF F=0 OR F=1 THEN LET FA=1
```

```
310 IF F=0 OR F=1 THEN GOTO 360
320 LET FA=F
330 LET FA=FA*(F-1)
340 LET F=F-1
350 IF F>1 THEN GOTO 330
360 RETURN
```

Voorbeeld

L
1.7
K
6
L = 1.7
K = 6
P = .0061243558

N. B. Het teken ** op regel 60 dient in één keer te worden ingetoetst. Dit teken staat boven de H-toets.

Fibonacci

De reeks van Fibonacci heeft het volgende uitgangspunt: iedere volgende term wordt gevonden door de twee voorafgaande termen op te tellen. Deze reeks heeft een aantal merkwaardige eigenschappen. Delen we bijvoorbeeld steeds de laatste twee termen, dan blijkt dit quotiënt meer en meer met de verhouding die onder de naam 'de gulden snede' bekend staat, overeen te komen.

Programma

```
10 PRINT "REEKS VAN FIBONACCI"
20 LET T1=1
30 LET T2=1
40 PRINT "VOER AANTAL TERMEN IN"
50 INPUT N
60 PRINT "T1=";T1
70 PRINT "T2=";T2
80 FOR K=3 TO N
90 LET TB=T2
100 LET T2=T2+T1
110 LET T1=TB
120 PRINT "T";K;"=";T2
130 NEXT K
```

Voorbeeld

```
REEKS VAN FIBONACCI
VOER AANTAL IN
7
T1 = 1
T2 = 1
T3 = 2
T4 = 3
T5 = 5
T6 = 8
T7 = 13
```

Integreren

Met het onderstaande programma kan men de integraal van een functie F(X) tussen de grenzen A en B berekenen.
Allereerst wordt het aantal stappen opgevraagd waarmee de integraal zal worden berekend. In het programma wordt gebruik gemaakt van de regel van Simpson.
De functie is in een subroutine ondergebracht: regel 300 t/m 320. In dit geval is als voorbeeld de functie X^3 d.w.z. X * X * X opgenomen.

Programma

```
10 PRINT "AANTAL STAPPEN"
20 INPUT N
25 PRINT N
30 PRINT "A="
40 INPUT A
45 PRINT A
50 PRINT "B="
60 INPUT B
65 PRINT B
70 LET S=(B-A)/N
75 LET SOM=0
80 FOR K=1 TO N-1
90 LET E=INT(K/2)-K/2
100 LET X=A+K*S
110 LET C=4
120 IF E=0 THEN LET C=2
130 GOSUB 300
140 LET SOM=SOM+C*F
150 NEXT K
160 LET X=A
```

```
170 GOSUB 300
180 LET FA=F
190 LET X=B
200 GOSUB 300
210 LET FB=F
220 LET J=S/3*(FA+SOM+FB)
230 PRINT "INT=";J
240 STOP
300 REM FUNCTIE
310 LET F=X*X*X
320 RETURN
```

Voorbeeld

AANTAL STAPPEN
20
A =
0
B =
2
INT = 4

N. B. In dit voorbeeld werd de integraal van de functie $F(X) = X^3$ uitgerekend, en wel tussen de grenzen 0 en 2. Wil men bijvoorbeeld de integraal van de functie $2X^2 + 3X$ uitrekenen (tussen de grenzen A en B) dan dient men alleen regel 310 te vervangen, nl. in:

310 LET F = 2 * X * X + 3 * X

Ringwerpen

We gooien met een ring naar een doel, dat zich op 9 meter afstand bevindt. Het doel is 1 meter groot. Iedere keer kunnen we de snelheid waarmee we gooien alsmede de hoek waaronder de ring wordt gegooid opgeven. Er zijn uiteraard maar drie mogelijkheden: de ring bereikt het doel niet, de ring schiet over het doel heen ... de ring treft het doel. De hoek dient steeds in graden opgegeven te worden. Zelfs de programmeur met onzekere hand zal het doel na een poosje zeer trefzeker kunnen raken!

<u>Programma</u>

```
10 PRINT "VOER SNELHEID IN"
20 INPUT V
30 PRINT "VOER HOEK IN"
40 INPUT A
50 CLS
60 LET A=A*2*PI/360
70 LET SA=SIN(A)
80 LET CA=COS(A)
90 LET T=V*SA/5+.2
```

```
100 LET S=V*CA*T
110 IF S>10 THEN PRINT "TE VER"
120 IF S<9 THEN PRINT "NIET VER GENOEG"
130 IF ABS(S-9.5)>=0.5 THEN GOTO 10
140 PRINT "GELUKT"
```

Voorbeeld

VOER SNELHEID IN
6
VOER HOEK IN
30
NIET VER GENOEG
VOER SNELHEID IN
13
VOER HOEK IN
45
TE VER
VOER SNELHEID IN
etc.

N. B. In regel 60 staat de term PI. Deze term moet met de M-toets (waar π bij staat) worden ingevoerd.

Maanlander

U zit in een maanlander 200 meter boven het maanoppervlak en uw daalsnelheid bedraagt -20 m/s. U kunt deze snelheid afremmen door de remraket te ontbranden. Dat kost natuurlijk wel brandstof (want zelfs op de maan geldt dat voor 'niets' alleen de zon opgaat). Iedere keer kunt u opgeven hoeveel brandstof verbruikt wordt. In het begin is er nog 100 liter in de tank. Na iedere keer meldt de computer de hoeveelheid brandstof, de daalsnelheid (v) en de hoogte (AFST). Goede landing!

Programma

```
10 LET V=-20
20 LET S=200
30 LET TA=100
40 PRINT "VOER BRANDST. IN"
50 INPUT BR
60 CLS
70 LET TA=TA-BR
80 LET A=BR-1
90 LET V=V+A
```

```
100 LET S=S+V-A/2
110 PRINT "BRANDST.=";TA
120 PRINT "V=";V
130 PRINT "AFST=";S
140 IF ABS(V)<.5 AND ABS(S)<.5 THEN GOTO 200
150 IF V>0 THEN PRINT "WE STYGEN"
160 IF TA<=0 THEN PRINT "TANK LEEG"
170 IF TA>=0 AND S>0 THEN GOTO 40
180 PRINT "MISLUKT"
190 STOP
200 PRINT "GELUKT"
```

Voorbeeld

```
VOER BRANDST. IN
5
BRANDST. = 95
V = -16
AFST = 182
VOER BRANDST. IN
5
BRANDST. = 90
V = -12
AFST = 168
VOER BRANDST. IN
etc.
```

Zoeken

ZOEKEN is een zeer eenvoudig spelletje. U dient steeds een X- en een Y-coördinaat op te geven om op deze wijze een onzichtbaar doel op te sporen. Als getallen mag u de getallen 1 t/m 20 invoeren. Het programma houdt precies bij hoeveel beurten nodig zijn om het doel te vinden. Na iedere opgave van een X- en Y-coördinaat meldt de computer zinvolle informatie, nl. of we meer naar rechts of links en tevens of we meer naar boven of naar onderen moeten zoeken.

Programma

```
10 RAND 0
20 LET P=INT(RND*20+1.0)
30 LET Q=INT(RND*20+1.0)
40 LET Z=1
50 PRINT "VOER X IN"
60 INPUT X
70 PRINT "VOER Y IN"
80 INPUT Y
90 LET Z=Z+1
100 CLS
110 IF P-X>0 THEN PRINT "NAAR RECHTS"
120 IF X-P>0 THEN PRINT "NAAR LINKS"
130 IF Q-Y>0 THEN PRINT "NAAR BOVEN"
140 IF Y-Q>0 THEN PRINT "NAAR BENEDEN"
150 LET A=(P-X)*(P-X)+(Q-Y)*(Q-Y)
160 IF A>0.1 THEN GOTO 50
170 PRINT "GELUKT IN ";Z;" BEURTEN"
```

Voorbeeld

VOER X IN
5
VOER Y IN

10

NAAR RECHTS
NAAR BOVEN
VOER X IN
etc.

Eeuwkalender

Dit programma bepaalt voor iedere datum van deze en de vorige eeuw de bijbehorende dag. Zo kunt u eindelijk eens nagaan op wat voor dag u geboren bent. Alle gegevens dienen als getal ingevoerd te worden. Zo heeft het getoonde voorbeeld betrekking op 30 augustus 1981.

Programma

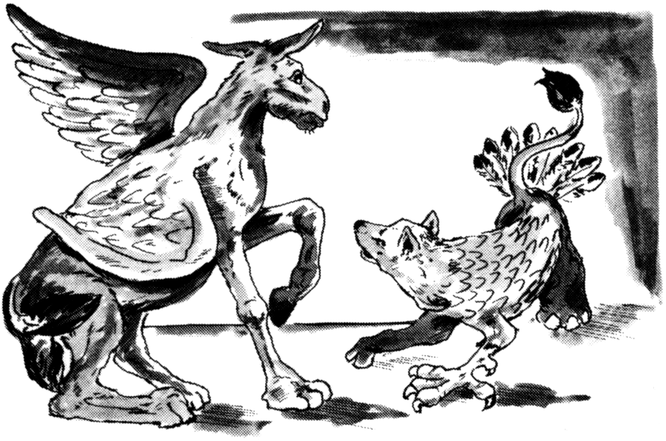
```
10 PRINT "DAG="
20 INPUT D
30 PRINT D
40 PRINT "MAAND="
50 INPUT M
60 PRINT M
70 PRINT "JAAR="
80 INPUT J
90 PRINT J
100 IF M<=2 THEN LET J=J-1
110 IF M<=2 THEN LET M=M+12
120 LET E=INT((13/5)*(M+1))
```

```
130 LET F=INT(5*J/4)
140 LET G=INT(J/100)
150 LET H=INT(J/400)
160 LET T=D+E+F-G+H
170 LET DA=INT((T/7-INT(T/7))*7+0.1)
180 PRINT "ZA=0, ZO=1, MA=2"
190 PRINT "DI=3, WO=4, DO=5"
200 PRINT "VR=6"
210 PRINT "DAG=";DA
```

Voorbeeld

```
DAG
30
MAAND
8
JAAR
1981
ZA=0, ZO=1, MA=2
DI=3, WO=4,DO=5
VR=6
DAG= 1
```

Dieren raden



Dit spel toont de mogelijkheden van de ZX81 als het gaat om manipulaties met letters. Iedere keer als u het commando RUN geeft, verschijnt een naam van een dier op het beeldscherm ... alleen de letters staan niet in de vertrouwde volgorde. Alle namen van de dieren zijn in feite in de string-variabele P$ verborgen. Deze is voor de overzichtelijkheid hieronder in een aantal delen verdeeld. Bij het intoetsen dient u alle letters direct na elkaar in te toetsen, dat wil zeggen er mogen geen spaties worden ingevoegd.

Programma

```
 5 RAND
 7 DIM A$(10,5)
 9 DIM B$(5,1)
11 LET G$=""
13 LET P$="LOHCSARBOCREDDA
           GEILVREGYTWUEEL
           LEREMKEONSSRAAB
           FARIG"
15 FOR K=1 TO 10
```

```
17 LET B1=(K-1)*5+1
19 LET A$(K)=P$(B1 TO B1+4)
21 NEXT K
23 LET R=INT(RND*10+1.)
25 FOR K=5 TO 1 STEP -1
27 LET B$(K)=A$(R,K)
29 LET G$=G$+B$(K)
31 NEXT K
33 LET X$=""
35 LET R=INT(RND*5+1.)
37 FOR K=R TO R+4
39 LET L=K
41 IF K>5 THEN LET L=L-5
43 LET X$=X$+B$(L)
45 NEXT K
47 PRINT X$
49 PRINT "IS EEN=?"
51 INPUT M$
53 CLS
55 IF M$<>G$ THEN PRINT "FOUT"
57 IF M$<>G$ THEN GOTO 47
59 PRINT "GELUKT"
```

Voorbeeld

YTREG
IS EEN = ?
TREYG
FOUT
YTREG
IS EEN = ?
TYGER
GELUKT

Trefwoordenlijst

A

aanhalingstekens 45, 84
aansluiten 14
aanzetten 14
ABS 58, 117
ACS 117
afronden 54
algorithme 27, 28
AND 64, 65, 117
argument 57
array 74
ASN 58, 117
AT 49, 50, 117
ATN 58, 117

B

beeldscherm 11
bewegende figuren 102
binair getal 111
bit 11
BREAK 61, 66 ev, 116
byte 11

C

calculator 38
cassetterecorder 106
cassettes 11, 13
centrale verwerkingseenheid 10
CHR$ 88, 117
CLEAR 39, 116, 125
CLS 49, 51, 102, 126
CODE 88, 118
codenummer 97, 132
commando 20, 26 ev, 38, 125
concateneren 84
CONT 61, 66 ev, 114, 116, 126
COPY 108, 126
COS 58, 118
CPU 10
cursor 24
G-cursor 97
K-cursor 24, 26
L-cursor 44
S-cursor 33, 113
> -cursor 25

D

decimale punt 19, 52
DIM 75, 126
dollarteken 83

E

E 55
END 19
EXP 58, 118
exponent 55

F

FAST 95, 102, 126
FOR ... NEXT 69, 71 ev, 127
foutboodschap 34, 113
functie 53, 57
functies 24
FUNCTION 24, 50

G

geheugen 10
geheugen-uitbreidingsmod. 11, 15
getallen 52
GOSUB 79, 127
GOTO 61, 62, 127
grafisch symbool 96
GRAPHICS 97

H

hardware 9 ev
herstellen van fouten 33 ev

I

IF ... THEN 61, 127
INKEY$ 95, 101, 118
INPUT 40, 43 ev, 56, 127
INT 53, 58, 118
instructies 125
integers 52
inverse 133

K

K 11
kolom-indeling 46, 49

L

LEN 85, 118
LET 41, 128
LIST 26, 128
listing 26, 128
LLIST 108, 128
LN 58, 118
LOAD 107, 128
LPRINT 108, 129
lijnen 11

M

machinecodes 10, 13
machine-instructies 109
machtsverheffen 58
mantisse 55
meer-dimensionale array 77
monitor 12

N

naam van variabele 18, 19, 20
NEW 26, 129
NEWLINE 25
NEXT 69, 71, 127, 129
NOT 64, 65, 118

O

Operating System 12
oplossingsmethode 27, 28 ev
OR 64, 118
overflow 114

P

PAUSE 95, 100, 129
PEEK 109, 111, 118
PI 59, 119
PLOT 87, 97
POKE 109, 110, 129
PRINT 43 ev, 129
PRINT AT 87, 94 ev
printer 11, 106, 108
prioriteit 30, 31
prioriteitsregels 123
programma 11, 13
programma-voorbeelden 135 ev

R

RAM 11
RAND 59, 130
reëele getallen 53
regelnummers 17, 26, 34
relatie-tekens 63
REM 43, 47, 130
RETURN 79, 130
RND 59, 119
ROM 11
RUBOUT 25, 35, 36
RUN 26, 130

S

SAVE 107, 130
schriftteken 83, 88
SCROLL 49, 51, 131
SGN 54, 58, 119
SIN 58, 119
SLICING 89 ev
SLOW 95, 102, 131
software 9, 12 ev
SPACE 25, 45
spaties 42
speciale woorden/termen 21, 22
SQR 57, 58, 119
STEP 72, 127
STOP 61, 63, 131
STR$ 87, 119
string-array 92
string-variabele 83
subroutines 79
symbolen 121, 132
syntax-fout 36

T

TAB 49 ev, 119
TAN 58, 119
teken v. getal 54
TO 72, 89
toetsenbord 22 ev
TV 14

U

uittesten 31
uitzetten 15
UNPLOT 87, 94, 102, 131
USR 109, 111, 119

V

VAL 86, 120
variabele 18
vermenigvuldigteken 19

W

wetenschappelijke notatie 53
wissen 25

Z

Z-80 10

Zelf programmeren is voor velen nog een wensdroom. De ZX81 (of ZX80 met uitgebreide BASIC-versie) brengt die mogelijkheid binnen ieders bereik.
Dit boek geeft een zeer duidelijke beschrijving van de mogelijkheden van de ZX81.
Er is een volledige BASIC-cursus in opgenomen waarbij juist gemikt is op gebruikers die geen of vrijwel geen ervaring met het programmeren hebben.
Tenslotte zijn veel voorbeeldprogramma's opgenomen.

ISBN 90 201 1515 4

ALBERT SICKLER ZX81